

Provable Secure Countermeasures Against Side-Channel Attacks

Vom Fachbereich Informatik der
Technischen Universität Darmstadt genehmigte

Dissertation

zur Erlangung des Grades
Doktor rerum naturalium (Dr. rer. nat.)

von

Clara Paglialonga, M.Sc.

geboren in Galatina (Italien).



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Referenten: Prof. Dr. Sebastian Faust
Prof. Dr. Stefan Dziembowski

Tag der Einreichung: 04.05.2020
Tag der mündlichen Prüfung: 19.06.2020

Darmstadt 2020

Author: Clara Paglialonga

Title: Provable secure countermeasure against side-channel attacks

Ort: Darmstadt, Technische Universität Darmstadt

Dieses Dokument wird bereitgestellt von tuprints,
E-Publishing-Service der TU Darmstadt
<http://tuprints.ulb.tu-darmstadt.de>
tuprints@ulb.tu-darmstadt.de

Bitte zitieren Sie dieses Dokument als:

URN:[urn:nbn:de:tuda-tuprints-185979](https://nbn-resolving.org/urn:nbn:de:tuda-tuprints-185979)

URI:<https://tuprints.ulb.tu-darmstadt.de/id/eprint/18597>

Die Veröffentlichung steht unter folgender Creative Commons Lizenz:

Attribution 4.0 International (CC BY 4.0)

<https://creativecommons.org/licenses/by/4.0/>



Erklärung

Hiermit versichere ich, die vorliegende Dissertation ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, Mai 2020

A handwritten signature in blue ink, reading "Clara Paglialonga", is written over a horizontal line.

(Clara Paglialonga)

Scientific career

January 2018 - June 2020

Continuation of the Promotion in Applied Cryptography under Prof. Dr. Sebastian Faust supervision at Technischen Universität Darmstadt, Germany

February 2016 - December 2017

Start of the Promotion in Applied Cryptography under Prof. Dr. Sebastian Faust supervision at Ruhr-Universität-Bochum, Germany

September 2012 - April 2015

Study of Master of Science in Mathematics at University of Turin, Italy

September 2009 - July 2012

Study of Bachelor of Science in Mathematics at University of Salento, Italy

List of Publications

- [FPS17] Sebastian Faust, Clara Paglialonga, Tobias Schneider “Amortizing Randomness Complexity in Private Circuits.” In: *Proceedings of Advances in Cryptology - ASIACRYPT 2017*, pages 781-810, Springer, 2017. **[Part of Chapter 3]**.
- [Bal+17] Josep Balasch, Sebastian Faust, Benedikt Gierlichs, Clara Paglialonga, François-Xavier Standaert “Consolidating Inner Product Masking.” In: *Proceedings of Advances in Cryptology - ASIACRYPT 2017*, pages 724-754, Springer, 2017. **[Part of Chapter 4]**.
- [Fau+18] Sebastian Faust, Vincent Grosso, Santos Merino Del Pozo, Clara Paglialonga, François-Xavier Standaert “Composable Masking Schemes in the Presence of Physical Defaults & the Robust Probing Model.” In: *Proceedings of IACR Transactions on Cryptographic Hardware and Embedded Systems - TCHES 2018, Volume 2018*, pages 89-120. **[Part of Chapter 5]**.
- [Sch+19] Tobias Schneider, Clara Paglialonga, Tobias Oder, Tim Güneysu “Efficiently Masking Binomial Sampling at Arbitrary Orders for Lattice-Based Crypto.” In: *Proceedings of Public-Key Cryptography - PKC 2019*, pages 534-564, Springer, 2019. **[Part of Chapter 6]**.
- [Bac+20] Florian Bache, Clara Paglialonga, Tobias Oder, Tobias Schneider, Tim Güneysu “High-Speed Masking for Polynomial Comparison in Lattice-based KEMs.” In: *Proceedings of IACR Transactions on Cryptographic Hardware and Embedded Systems - TCHES 2020, Volume 2020*.

Acknowledgments

This thesis contains the result of my years of research at the Chair of Applied Cryptography, first at Ruhr-Universität Bochum and then at Technische-Universität Darmstadt. These years deeply signed my life development, both as a researcher and more globally as a person, thanks to the contribution of the people I met on my path.

First of all, I want to express my gratitude to my supervisor Sebastian Faust for believing in my capabilities to pursue a Ph.D. Thanks for all the fruitful discussions and useful advice, and thanks for leaving me high independence in my projects.

Secondly, I would like to thank my disputation committee, and especially my external referee Stefan Dziembowski, for the time they spent reading my thesis and the interesting questions and feedback they provided me. For the financial support, my acknowledgments go to the DFG and the BMBF, which provided me with the essential resources for my studies. I am very grateful to Marion Reinhardt-Kalender, Anja Krause, Andrea Püchner and Jacqueline Wacker for their extremely helpful support about bureaucratic university processes.

My research results would have not been the same without the collaboration and the exchanges I had with my co-authors. I thank all of them for sharing productive discussions and sleepless nights before deadlines. Among them, I am particularly grateful to Tobias Schneider, who has always been there whenever I asked him for help.

A special acknowledgment goes to all my Ph.D. fellows, who impacted these four years in Bochum and Darmstadt with their diverse personalities. Thanks to those who have shared with me long chats about life, open-air lunch breaks at the botanical garden, road trips after conferences, thanks to those who have been with me until the very end of social events, and especially thanks to everyone for accepting my moody spirit and my constant jokes. In particular, I am very glad to have shared the entire path of my Ph.D. with Lisa Eckey and Kristina Hostáková. I have deep gratitude for their endless support and unconditional kindness, and I am happy to have found authentic friends in them.

The last years have been particularly intense not only for all opportunities I had about my studies but also because I experienced living as an ex-pat in Germany. I met a countless number of people from different parts of the world, some of them just for a hike, some of them for much more. Without them, my personal growth would have not reached the same point and I am extremely grateful to all who have shared with me their traditions, points of view, and life stories contributing to broadening my horizons. I thank in particular all my new friends and my flatmates, who have represented an exit

way from stress: from bouldering to playing guitar, from doing DIY to biking, the list of things I loved doing together could be infinite.

Next to them, I am grateful the distance did not separate me from my old friends in Italy, who constantly encouraged me even from far. Among them, I want to particularly thank Davide Cois. Despite his total inexperience in the field of cryptography, he has been next to me in many key moments of the path that brought me to pursue my Ph.D.

During the intensive months of writing this thesis, I'm thankful I had the important support of my partner, who patiently bore my stressed condition and motivated me to keep my tight work schedule of the last weeks.

Last but not least, my acknowledgments go to my parents, who always accepted and supported my choices, even when they would have brought me thousands of kilometers away from them, and to my sister, who, more than everyone, has been always with me despite any physical distance.

Contribution

The results of this thesis are the outcome of collaborations and cooperative works with my supervisor, Sebastian Faust, and my co-authors, Josep Balasch, Benedikt Gierlichs, Vincent Grosso, Tim Güneysu, Tobias Oder, Santos Merino Del Pozo, Tobias Schneider, and François-Xavier Standaert, with whom I shared countless hours of discussions. The highly collaborative nature of the research process sometimes makes it difficult to break down and associate particular components of each paper to individual authors. Nevertheless, in this section, I give, where possible, an explicit description of my contribution to the results included in this thesis. In general, in all my publications, I have been responsible for the theoretical contribution and the study of the security analysis. More details follow.

The results from Chapter 3 are based on the publication “Amortizing Randomness Complexity in Private Circuits” [FPS17], which was realized in collaboration with Sebastian Faust and Tobias Schneider. The construction presented is the result of an equal collaboration with my co-authors, while the formalization and security analysis constitute my contribution entirely. The implementation was done by Tobias Schneider.

The results from Chapter 4 are based on the publication “Consolidating Inner Product Masking” [Bal+17], which was realized in collaboration with Josep Balasch, Sebastian Faust, Benedikt Gierlichs and François-Xavier Standaert. The algorithms introduced in the paper were developed in collaboration with Sebastian Faust, while their security analysis is exclusively my contribution. The information theoretic evaluation was performed by François-Xavier Standaert while the performance evaluations and the empirical side-channel leakage evaluation by Josep Balasch and Benedikt Gierlichs.

The results from Chapter 5 are based on the publication “Composable Masking Schemes in the Presence of Physical Defaults & the Robust Probing Model” [Fau+18], which was realized in collaboration with Sebastian Faust, Vincent Grosso, Santos Merino Del Pozo, and François-Xavier Standaert. In this paper, Sebastian Faust, Vincent Grosso, François-Xavier Standaert, and me jointly participated to the conceptual part. My contribution consisted of guiding the theoretical part and writing the security proofs. Santos Merino Del Pozo and François-Xavier Standaert contributed to the practical experiments.

The results from Chapter 6 are based on the publication “High-Speed Masking for Polynomial Comparison in Lattice-based KEMs” [Sch+19], which was realized in collaboration with Tobias Schneider, Tobias Oder, and Tim Güneysu. In this paper, Tobias

Schneider and me jointly developed the algorithmic part, to which I particularly contributed providing the security analysis. Tobias Oder did mainly the implementations. Tim Güneysu helped us with his experience in lattice-based cryptography.

Abstract

Side-channel attacks are a prominent threat to the security of cryptographic implementations. Differently from the traditional black-box attacks, which exploit the inputs and outputs of cryptographic schemes, side-channel attacks partially access the inner working of the scheme as well, by observing the physical leakage emitted by the device executing cryptographic algorithms. A notable example is the class of *power-analysis attacks*, which exploits the power consumption of the underlying device to recover the secret keys of the implemented cryptosystem.

Since their first presentation in the late 1990s, the problem of securing cryptographic systems in the presence of side-channel leakages received significant attention by the cryptographic community. In particular, the theory community has concentrated its research efforts on formally modeling the side-channel leakages and on designing cryptographic schemes *provably* secure in such leakage models. However, conceiving a formal model that realistically captures possible physical leakages and constructing primitives sufficiently efficient to be deployed in practice are still challenging research objectives. This research area is usually referred to as leakage-resilient cryptography.

In this dissertation, we examine three main research directions. The first one focuses on the technique of masking, which is the most popular approach to counteract power-analysis attacks. In this context, our work analyzes possible methods to improve the efficiency of masked implementations, with the goal of overcoming the decrease in performance, which is a common drawback of masking. In particular, we provide new and more efficient algorithms for the computation of basic operations in two kinds of masking schemes: the Boolean masking and the Inner Product masking.

The second research direction of this work addresses the problem of formalizing security requirements for hardware implementations. While provable security of software-oriented masking has been studied since the origin of leakage resilient cryptography, the more complex task of formally proving the security of hardware-oriented masking is receiving attention by the community only recently. We discuss the different challenges of this field, and we introduce a new theoretical model, the robust probing model, capturing the conditions required for securely implementing complex algorithms in practice, providing a guide to practical implementers.

Finally, the last part of this work is concerned with the protection of algorithms employed in lattice-based constructions, which are studied in the field of post-quantum cryptography. While several works investigate the application of masking schemes to

standard cryptosystems, for most post-quantum schemes this is still an ongoing research. We contribute to this research area, by applying the masking countermeasure to a binomial sampler which is at the base of many lattice-based cryptosystems.

Zusammenfassung

Seitenkanalangriffe sind eine erhebliche Bedrohung für die Sicherheit von kryptographischen Implementierungen. Anders als bei den traditionellen Black-Box-Angriffen, die nur die Eingaben und Ausgaben von kryptographischen Verfahren betrachten, greifen Seitenkanalangriffe auch auf die inneren Abläufe zu, indem sie physikalisches Leakage beobachten, das von dem Gerät ausgegeben wird, während es kryptografische Algorithmen ausführt. Ein bekanntes Beispiel ist die Klasse der Power-Analyse-Angriffe, bei der der Stromverbrauch des zugrundeliegenden Geräts gemessen wird, um die geheimen Schlüssel des implementierten Kryptosystems wiederherzustellen. Seit der Einführung in den späten 90er Jahren, erfährt das Problem der Absicherung kryptografischer Systeme gegenüber Seitenkanalanalysen erhebliche Aufmerksamkeit in der kryptografischen Forschungsgemeinschaft. Insbesondere im Forschungszweig der theoretischen Kryptographie sind Anstrengungen unternommen worden, Leakage durch Seitenkanäle formal zu modellieren und beweisbar sichere kryptografische Schemata zu entwickeln. Die Konzeption eines formalen Modells, das mögliches Leakage realistisch erfasst sowie die Konstruktion von sicheren Primitiven, die ausreichend effizient für den praktischen Einsatz sind, sind nach wie vor anspruchsvolle Forschungsziele. Dieser Forschungsbereich wird üblicherweise als Leakage-Resiliente Kryptographie bezeichnet. In dieser Dissertation werden drei Hauptforschungsrichtungen untersucht. Die erste konzentriert sich auf die Technik des „Masking“, welche der beliebteste Ansatz ist, um Power-Analyse-Angriffen entgegenzuwirken. In diesem Zusammenhang analysiert diese Arbeit mögliche Methoden zur Verbesserung der Effizienz von Masking-Implementierungen, mit dem Ziel, den Leistungsabfall zu überwinden, der ein häufiger Nachteil solcher Verfahren ist. Insbesondere bieten wir neue und effizientere Algorithmen für die Berechnung von grundlegenden Operationen in zwei Arten von Masking-Verfahren: das „Boolesche Masking“ und das Skalarprodukt-Masking. Die zweite Forschungsrichtung dieser Arbeit befasst sich mit dem Problem der Formalisierung von Sicherheitsanforderungen für Hardware-Implementierungen. Die beweisbare Sicherheit von software-orientiertem Masking wird seit der Einführung der Leakage-Resilienten Kryptographie betrachtet, jedoch wurde der komplexeren Aufgabe der beweisbaren Sicherheit von hardware-orientiertem Masking erst in jüngster Zeit Aufmerksamkeit geschenkt. Wir diskutieren die verschiedenen Herausforderungen in diesem Bereich und stellen ein neues theoretisches Modell vor, das sogenannte „robuste Probing-Modell“, welches die Bedingungen für sichere Implementierung komplexer Algorithmen in der Praxis erfasst und einen Leitfaden für

Programmierer darstellt. Der letzte Teil dieser Arbeit befasst sich schließlich mit dem Schutz von Algorithmen, die in Gitter-basierten Konstruktionen eingesetzt werden, welche im Bereich der Post-Quantum-Kryptographie untersucht werden. Obwohl mehrere Arbeiten die Anwendung von Masking-Verfahren auf Standard-Kryptosysteme untersuchen, zählt dies für die meisten Post-Quantum- Schemata noch zur laufenden Forschung. In dieser Dissertation untersuchen wir die Maskierung eines Algorithmus zur Erzeugung der Binomial-Verteilung, die vielen Gitterbasierten-Verfahren zugrunde liegt.

Contents

1. Introduction	1
1.1. Attacking a cryptosystem	2
1.2. Goals of the thesis	5
1.3. Organization of the thesis	7
2. Background	9
2.1. Notation	9
2.2. Recovering secrets: Power Analysis attacks	9
2.2.1. Simple Power Analysis attack	10
2.2.2. Differential Power Analysis attack	11
2.2.3. Profiling attacks	14
2.3. Leakage models	14
2.4. Masking, a countermeasure against SCA	17
2.5. Conversion algorithms between Arithmetic and Boolean masking	18
2.6. Proving security in the probing model	20
2.7. Threshold Implementation	24
2.8. Cryptographic building blocks	25
2.8.1. A symmetric key encryption scheme: the AES	25
2.8.2. A lattice-based key exchange protocol: the NEWHOPE	27
3. Randomness optimization in Boolean masking	31
3.1. Reusing randomness in private circuits	36
3.2. A t – SCR multiplication Scheme	41
3.3. A t – SCR refreshing scheme	59
3.4. First-order security with constant amount of randomness	62
3.5. Case study: AES	67
3.6. Conclusions	71
4. Randomness optimization in Inner Product masking	73
4.1. A new multiplication scheme for IP masking	78
4.2. Application to AES S-box	84
4.2.1. A more efficient scheme for dependent inputs	84
4.3. Performance evaluations	90

4.4. Information theoretic evaluation	91
4.5. Empirical side-channel leakage evaluation	93
4.6. Conclusions	95
5. Masking hardware oriented implementations: the robust probing model	97
5.1. A composability notion for Threshold Implementation	102
5.1.1. Pseudo-NI and pseudo-SNI security	102
5.1.2. The tradeoff between number of shares and cycle count	104
5.2. The robust probing model	105
5.3. Physical defaults combination	109
5.3.1. Experimental validation	111
5.4. Concrete constructions	112
5.4.1. The Toffoli gate is pseudo-(1,0,0)-robust 1-probing secure . . .	113
5.4.2. The ISW is (1,0,0)-robust t -SNI with $t+1$ shares in 2 cycles . .	114
5.4.3. Glitch locality principle	118
5.5. Practical security evaluation	118
5.6. Conclusions	120
6. Masking lattice-based cryptographic primitives	121
6.1. New Conversion Algorithms from Boolean to Arithmetic masking	125
6.1.1. Improved higher-order $B2A_q$ from [Bar+18b]	125
6.1.2. A new $B2A_q$ conversion algorithm for $x \in \mathbb{F}_2$	129
6.1.3. A new $B2A_q$ conversion algorithm for $x \in \mathbb{F}_{2^k}$	135
6.1.4. Performance analysis	137
6.2. Masked implementation for higher-order binomial samplers	138
6.2.1. Generalization at high-order of [Ode+18]	139
6.2.2. New bit-sliced masked binomial sampler	141
6.3. Case study: NEWHOPE	145
6.4. Conclusions	145
7. Conclusion	147
A. Appendix	149
A.1. Additional algorithms used in Chapter 6	149
Bibliography	153

List of Abbreviations

AES	Advanced Encryption Standard.
B2A	Boolean to Arithmetic.
B2A_q	Boolean to Arithmetic modulo q .
BC	Before Christ.
CCA	Chosen Ciphertext Attack.
CPA	Chosen Plaintext Attack.
DES	Data Encryption Standard.
DPA	Differential Power Analysis.
FPGA	Field-Programmable Gate Array.
HW	Hamming Weight.
ID	Identification.
KEM	Key Encapsulation Mechanism.
LSB	Least Significant Bit.
LWE	(decisional) Learning With Errors (problem).
MAC	Message Authentication Code.
MIA	Mutual Information Analysis.
NI	Non Interference.
NIST	National Institute of Standards and Technology (United States).
PPT	Probabilistic Polynomial-Time.
PRF	Pseudo-Random Function.
PRNG	Pseudorandom Number Generator.
RAM	Random-Access Memory.
RNG	Random Number Generator.
RSA	Rivest-Shamir-Adleman.

SCR	Security with Common Randomness.
SIM	Subscriber Identity Module.
SNI	Strong Non Interference.
SPA	Simple Power Analysis.
TI	Threshold Implementation.
TRNG	True Random Number Generator.
XOR	Exclusive OR.

List of Algorithms

2.1. Addition algorithm with $n \geq 2$ shares	23
2.2. ISW multiplication algorithm with $n \geq 2$ shares	23
2.3. Multiplicative refreshing scheme	24
2.4. TI multiplication algorithm for $t = 1$	25
2.5. NEWHOPE CPA.Keygen	28
2.6. NEWHOPE CPA.Encryption	28
2.7. NEWHOPE CPA.Decryption	29
3.1. Refreshing gadget Ind	40
3.2. Mult^2 for order $t = 2$ with $n = 3$ shares	43
3.3. Mult^3 for order $t = 3$ with $n = 4$ shares	44
3.4. Mult^4 for order $t = 4$ with $n = 7$ shares	44
3.5. Mult^5 for order $t = 5$ with $n = 7$ shares	45
3.6. Refreshing scheme \mathcal{R}' for order $t = 4$	60
3.7. SecMult case (i)	62
3.8. Refreshing case (i)	63
3.9. SecMult case (ii) and (iii)	64
3.10. SecMult case (iv), (v) and (vi)	64
3.11. Modified refreshing \mathcal{R}'	65
3.12. Modified addition Add'	66
4.1. Setup the masking scheme: $\mathbf{L} \leftarrow \text{IPSetup}_n(\mathcal{K})$	79
4.2. Masking a variable: $\mathbf{S} \leftarrow \text{Encode}_{\mathbf{L}}(S)$	80
4.3. Add masked values: $\mathbf{C} \leftarrow \text{Add}(\mathbf{A}, \mathbf{B})$	80
4.4. Refresh vector: $\mathbf{X}' \leftarrow \text{IPRefresh}_{\mathbf{L}}(\mathbf{X})$ [BFG15]	81
4.5. Refresh vector: $\mathbf{Y} \leftarrow \text{SecIPRefresh}_{\mathbf{L}}(\mathbf{X})$	81
4.6. Multiply masked values: $\mathbf{C} \leftarrow \text{IPMult}_{\mathbf{L}}^{(1)}(\mathbf{A}, \mathbf{B})$	82
4.7. Square masked variable: $\mathbf{Y} \leftarrow \text{IPSquare}_{\mathbf{L}}(\mathbf{X})$	84
4.8. Multiply dependent masked values: $\mathbf{C} \leftarrow \text{IPMult}_{\mathbf{L}}^{(2)}(\mathbf{A}, g(\mathbf{A}))$	86
5.1. Modified ISW multiplication algorithm with $n \geq 2$ shares	115
6.1. SecBoolArithModp [Bar+18b]	126

6.2. SecArithBoolModp (cubic) [Bar+18b]	127
6.3. SecArithBoolModp (quadratic)	127
6.4. SecB2A _{q-Bit} (simple)	130
6.5. SecB2A _{q-Bit} (optimized)	131
6.6. B2A _{q-Bit}	131
6.7. B2A ⁽ⁿ⁾ _{q-Bit}	132
6.8. RefreshADD (based on RefreshXOR [Bar+18b])	135
6.9. SecB2A _q	135
6.10. SecSampler ₁	141
6.11. SecBitAdd	142
6.12. SecBitSub	143
6.13. SecConstAdd	143
6.14. SecConstAdd (optimized for $\kappa = 8$)	144
6.15. SecSampler ₂	144
A.1. SecAddModp [Bar+18b]	149
A.2. SecAdd [Bar+18b]	150
A.3. SecAnd [CGV14]	150
A.4. FullXOR [CGV14]	150
A.5. RefreshXOR [Bar+18b]	151
A.6. FullRefreshXOR [Bar+18b]	151
A.7. SecMul (based on SecAnd)	151

List of Figures

2.1. Set-up of a power analysis attack	10
2.2. Power consumption traces of an implementation of AES	11
2.3. Example of differential curve for incorrect hypothesis of the key	12
2.4. Example of differential curve for correct hypothesis of the key	12
2.5. DPA principle using a prediction model	13
2.6. Inversion chain of the AES S-box	27
3.1. Example of two gadgets sharing randomness	33
3.2. Example of two blocks of gadgets sharing randomness	34
3.3. Example of randomness reuse at 1 st order	35
3.4. A set of N blocks of gadgets with dimension $d = 4$	38
3.5. Example of different use of the <code>lnd</code> refreshing scheme	41
3.6. Example of composition where the randomness cancels out	67
3.7. AES S-box with the insertion of the <code>lnd</code> gadget.	68
4.1. Multiplication scheme by [BFG15]	75
4.2. New <code>IPMult</code> ⁽¹⁾ for $t = 2$	76
4.3. New <code>IPMult</code> ⁽²⁾ for $t = 2$	77
4.4. Gadget \cdot^{254} using <code>IPMult</code> _L ⁽¹⁾ and <code>IPMult</code> _L ⁽²⁾	88
4.5. IT evaluation of IP encoding with HW leakage	92
4.6. IT evaluation of IP encoding with random leakage	93
4.7. t-test for Boolean and IP masking with RNG deactivated	94
4.8. t-test for Boolean and IP masking with RNG activated	95
5.1. (1,0,0)-robust 2-SNI implementation of ISW	100
5.2. Decomposition example of a Toffoli gate	106
5.3. Example of physical defaults in TI	107
5.4. Combination of transitions and glitches	110
5.5. t-test results for a 1 st order TI of PRESENT S-box	112
5.6. t-test results for a 1 st order TI of PRESENT S-box	113
5.7. t-test of PRESENT S-box for 2 shares	119
5.8. t-test of PRESENT S-box for 3 shares	119
6.1. Structure of the new bit-sliced sampler	123

6.2. Structure of the new $B2A_q$, for q prime and $t = 2$	123
6.3. Recurrent scheme in Algorithm 6.3	128
6.4. Structure of SecArithBoolModp for $n = 4$	129
6.5. Structure of $\text{SecB2A}_{q-\text{Bit}}$	132
6.6. Structure of SecB2A_q $k = 3$	136
6.7. Structure of SecSampler_2	142

List of Tables

3.2.	Randomness comparison with and without randomness reuse	69
3.3.	Cycle counts comparison of AES implementations	70
4.1.	Complexity comparison of $\text{IPMult}_{\mathbf{L}}^{(1)}$ and $\text{IPMult}_{\mathbf{L}}^{(2)}$	89
4.2.	Performance evaluation of IP masked AES-128	91
5.1.	Occurances of an output for a given input in a Toffoli TI gate	104
6.1.	Operation count for B2A conversions modulo $q = 12289$	138
6.2.	Operation count for B2A conversions modulo 2^k	139
6.3.	Randomness count for B2A conversions with $q=12289$	139
6.4.	Randomness count for B2A conversions with modulo 2^k	140
6.5.	Cycle counts for masked sampling	146

1. Introduction

Securing information has been a need since the start of civilization. Today, with the advent of digital communication, more and more sensitive data circulates every day through the network. On top of that, the systems for processing and storing information are constantly evolving. The digital devices that are increasingly becoming part of our daily life are countless. Prominent examples are smart cards, that have applications in various fields, like identification systems (ID badges), financial exchange (credit cards), mobile phones (SIM), health-care (insurance cards), etc. Consequently, protecting data from unauthorized access is becoming an increasingly complex task, as it concerns both the digitalization of the information and the physical devices which process it.

Cryptography is a discipline that designs methodologies and techniques to protect secrets. It is at the heart of today's communication and technology, and its development started in ancient times, presumably in 1900 BC, when Egyptian scribes used a non-standard fashion to write hieroglyphs in an inscription on the wall of a tomb [Kah96]. As the history of cryptography is long, its objectives and techniques significantly evolved with time. Classical cryptography was characterized by heuristic approaches and it manipulated alphabetic characters. It was the method of choice until late in the 20th century and, as pointed out by [KL14], it was more appropriately “an art than a science”. With the beginning of the digital era, cryptography evolved into a proper science, after called modern cryptography, where algorithms operate on a sequence of binary bits and the security analysis is characterized by the use of formal methods and mathematical algorithms, commonly called primitives. The following properties can summarize its fundamental objectives: *confidentiality*, i.e., protecting information from the access of unauthorized parties; *data integrity*, i.e., protecting information from unauthorized modifications; and *authentication*, i.e., correctly identifying communicating parties and data origin.

In modern cryptography, we define a *cryptosystem* as a set of cryptographic algorithms needed in order to fulfill a particular security goal. A typical example is the cryptosystem for encryption. The scheme consists of three probabilistic polynomial-time (PPT) algorithms: a *key-generation algorithm*, which on input 1^κ , where κ is the security parameter, outputs a key; an *encryption algorithm*, which transforms a plaintext, i.e., the information to protect, into a *ciphertext*; and a deterministic *decryption algorithm*, which, using the key, recovers the plaintext from the ciphertext (or outputs an error).

The different way in which the keys are used in the algorithms divides modern cryptography into two main branches: *symmetric cryptography* and *asymmetric cryptography*. In symmetric cryptography, the parties involved share a common secret key, which they use to communicate securely. These kinds of cryptographic algorithms are the ones that have been in use since historic times. Asymmetric cryptography is instead a more recent branch, which has increased in popularity after the seminal work of Diffie-Hellman [DH76]. In asymmetric cryptography, no common secret is needed, and the exchange of sensitive information between two entities relies on algorithms that make use of key pairs, one for each party. The first key is public, and it is available to anyone who wants to use the algorithm securely. The second key is, instead, kept private by each party, and it is used to read the piece of information secretly received. Examples of primitives in symmetric cryptography are the message authentication codes (MACs), and the private-key encryption schemes. Relevant kinds of private-key encryption schemes are block ciphers, which process the plaintext in groups of bits, also called blocks. Most influential block ciphers have been DES [Des] and AES [RD01]. While the former one is considered insecure due to its small key size, the latter one is one of the most popular cryptosystems. In asymmetric cryptography, on the other hand, the related primitives are public-key encryption schemes and digital signatures. In both symmetric and asymmetric cryptography, private keys play a central role in guaranteeing security. Any entity who wants to defeat the goals of a cryptosystem, commonly called an *attacker*, would not be able to reach his intent without the knowledge of the secret key.

In the next section, we focus on the models that formalize the possible strategies that an attacker can use to defeat a cryptosystem, and we take into particular consideration the attacks on cryptographic implementations.

1.1. Attacking a cryptosystem

An essential aspect of modern cryptography is that cryptographic schemes are considered secure only after their security requirements are well defined and formally proven. This concept is called *provable security* and provides high confidence in the security of modern cryptography. At the base of provable security there is the definition of a *security model*, which consists of two components: the *adversarial model*, i.e., the kind of abilities the adversary has in order to attack, and the *security requirements*, i.e., the properties that an implementation needs to fulfill in order to be defined secure.

More precisely, when an adversary targets a cryptosystem, he assumes various approaches according to the different weaknesses of the algorithm that he decides to exploit and the different goals he wants to achieve. Therefore, to show that a cryptographic scheme is secure, the cryptographer first has to define an adversarial model, which formally describes the goals, assumptions, and capabilities of a particular attacker. Some

common adversarial models are the *chosen-plaintext attack* (CPA), and the *chosen-ciphertext attack* (CCA), where the adversary targets the encryption scheme with the goal of distinguishing the encryption of two messages of the same length. In particular, in a CPA, a PPT adversary has the ability to choose two messages of the same length and learn the encryption of one of them arbitrarily. His goal is to guess which message corresponds to the encryption received. In a CCA, a PPT adversary can additionally obtain the decryption of ciphertexts of its choice, except for the one previously received as challenge. Other usual adversarial models are the ones on digital signature schemes, like the *known-message attack*, *chosen-message attack*, and *adaptive chosen-message attack* where the adversary's goal is to make a forgery of the signature.

Once an adversarial model is specified, a formal proof demonstrates that a scheme is secure against the attacker defined in the model. All the adversarial models listed above are generally denoted as *black-box attacks*, and most cryptosystems have been proven secure against these kinds of attacks. Shafi Goldwasser and Silvio Micali pioneered this area with a large amount of work, demonstrating for several classes of primitives that they can be formally proven secure in different adversarial models. Examples are their analysis in the fields of encryption [GM82; GM84] and signatures [GMR84; GMR88].

Many security proofs are based on the fact that an attacker can break the cryptosystem only by solving a computationally infeasible problem. For example, the hardness of factoring a large integer is at the base of the security proof of the famous RSA encryption scheme, or the discrete logarithm problem is at the core of several schemes, like the ElGamal encryption scheme and the Diffie-Hellman key exchange protocol. In this case, the proof of security is called a *reduction* and, according to the adversarial model, shows that the cryptosystem meets the security requirements when the hypotheses about the adversary's abilities are satisfied, and determined assumptions about the hardness of specific computational tasks hold. We underline that, if an attacker has a significant amount of time and computational power, he can always perform a straightforward type of attack, the *brute force attack*, which works by exhaustively trying to decrypt the encrypted message with all possible secret keys.

Quantum attacks

From the 1990s, the introduction of the concepts of *quantum computing* and *quantum computers* showed that certain computational complexity assumptions used in security proofs can be broken. In particular, Peter Shor published in 1997 an algorithm which solves the integer factorization problem in polynomial time using quantum computation [Sho97]. As a consequence, some of the cryptographic schemes considered secure nowadays might be unreliable when quantum computers will be available. At the moment, the examples of existing quantum computers are few [IBM17; Kel18]. However some researchers and governance have predicted that quantum computers capable of breaking currently used

cryptosystems could be developed by the 2030s [Mos15; Tou+16].

In order to guarantee a secure transition towards quantum computing, the branch of *post-quantum* cryptography has emerged, with the promise of developing cryptographic alternatives to today's schemes provable secure against attackers with quantum computing power. The National Institute of Standards and Technology (NIST) has recently run a standardization process for post-quantum cryptography [ST17], which received significant attention. *Multivariate*-, *code*-, *hash*-, *isogeny*-, and *lattice-based* cryptographic schemes [BBD09; FJP14] are the most prominent examples of solutions that guarantee the black box security and additionally provide post-quantum protection. They base their approaches on hardness assumptions that are supposedly unbreakable by quantum computers in polynomial time.

Attacks on cryptographic implementations

In this thesis, we mainly focus on less classical attacks, namely the attacks on the implementation of a cryptographic system. Modern cryptography is suited to work on binary fields, and it is implemented on devices that process binary representations. Implementations can be performed in software or hardware. An attacker on cryptographic implementations, instead of exploiting the functionality of a cryptosystem, profits from weaknesses in its implementation, e.g., in the software code or in the physical device.

More generally, following the classification from [MOP08], two types of attackers can be distinguished. First, we can have *active attacks* and *passive* ones, depending on whether the attacker interacts with its target or not. The active attacks alter the information of the cryptosystem by, for example, modifying some data unauthorizedly. On the other hand, a passive attacker adopts techniques that allow him to access sensitive information via observing the execution of the protocol only and without influencing it. A second criterion of classification is based on the level of intrusion into the target and distinguishes *non-invasive*, *semi-invasive*, and *invasive attacks*. In a non-invasive attack, an attacker exploits directly accessible interfaces of the target device and accesses only the environmental parameters, without altering the device itself. On the contrary, in an invasive attack, the attacker reaches full control of the target device by using specialized equipment, which allows, for instance, to establish electrical contact with the chip surface. Semi-invasive attacks present a more moderate level of intrusion in the target device, like removing the packaging of a microchip in order to facilitate access to inner components of the device.

Non-invasive attacks are the most common implementation attacks, since they are cheaper to perform. In this context, a typical active attacker aims at influencing the execution of the cryptosystem by modifying some wires of the circuit where the protocol is implemented, as in the so-called fault attacks [BDL97; BS97; Ish+06; KJJ99]. On the other hand, a passive attacker can, for instance, learn sensitive information by analyzing

physical properties of the hardware implementation, like the device’s emanations or power consumption during an encryption operation [Agr+03; KJJ99], or the time needed to perform it [Ald+19; Koc96]. The attacks that make use of the physical leakages of a cryptographic implementation are called *side-channel attacks* (SCA in short) and they are the focus of our work.

The implementation of a cryptographic scheme can be formally proven secure against physical attacks as well. As these attacks are outside of the mathematical models, it is necessary to extend the standard adversarial models and to define new security models, which formalize the physical attack in such a way that it can be formally analyzed. In the context of side-channel attacks, a security model is called *leakage model*, and its main challenge is specifying realistically the physical leakages that can affect an implementation. A cryptographic implementation secure in such a model is said to be *leakage resilient*.

1.2. Goals of the thesis

In this thesis, we intend to address the problem of securing cryptographic algorithms when implemented in practice, and in particular against the so-called *power analysis attacks*. Our work focuses on analyzing one of the main countermeasures against such attacks, the so-called technique of *masking*. A cryptographic algorithm protected under such a technique is said a *private circuit* [ISW03]. Our analysis investigates some of the main challenges of masking.

The first challenge is finding a good trade-off between efficiency and security. The latter one, indeed, often comes at the cost of reduced performances, and it is the main obstacle in the adoption of cryptography in practice. We decided to tackle one of the major drawbacks of masking schemes, which is the blow-up in the complexity of the masked version of a circuit, compared to the unmasked one. The process of masking causes an increase in the number of operations to perform. Additionally, it makes extensive use of random elements, which are usually outputs of devices or algorithms, like TRNGs or PRNGs. The generation of randomness and the shipment to the place where it is needed is very costly, and one of the main challenges of practical implementation of masking. To give an example, the masked implementation of the AES block cipher using Boolean masking, at the minimum security level, makes use of 1,200 calls to the TRNG, and this number increases with the security level. We noticed that this amount of randomness can be reduced. While other works are trying to reach this goal by reducing the use of random bits within the computations of the primary circuit operations [Bel+16; Bel+17], we investigated a different approach, which consists of recycling some of the randomness used by the circuit. Additionally, we analyzed a type of masking scheme which shows a high-security level, the so-called *Inner Product masking* [Bal+12; BFG15; DF12; GR12]. In this context, we consolidated the works of Balasch *et al.* [Bal+12;

BFG15]. In particular, on a conceptual level, we improved the existing algorithms, by simplifying the schemes in order to reduce their use of randomness. We provided algorithms that are more efficient and easy to implement. The algorithmic results are enriched by new implementation results, the information theoretic and the experimental evaluations, that have been conducted in [Bal+17] and that complete the theoretical and practical understanding of this masking scheme.

Another relevant aspect for applying private circuits in practice is the development of security models that are closer to reality. The literature abounds of leakage models that formalize the power analysis attacks and the conditions under which a circuit is secure. They differ from each other by how close they match reality and ease of usage in proofs. We formalized our works mainly in the so-called *probing model* [ISW03], which is one of the most popular leakage models, thanks to its highly simplified structure, which allows the development of intuitive proofs and algorithms. Unfortunately, the probing model lacks security guarantees against further physical effects that can occur in hardware implementations. In this setting, the leakage can recombine, for instance, during consecutive use of registers or due to differences in timing signals, causing a reduction of the security level. Therefore we studied the problem of tweaking the popular probing model with the intent of capturing a broad class of physical effects, and present the *robust probing model*.

Lastly, we contributed to the protection of post-quantum cryptography against power analysis attacks, by applying the masking countermeasure to lattice-based cryptography. While for common symmetric and asymmetric cryptographic schemes the application of masking schemes has been largely analyzed and achieves a reasonable level of security and efficiency, this is still ongoing research for most post-quantum schemes. Among the 69 complete submissions to the NIST competition, lattice-based cryptography represents the biggest share of them, with a total of 29 submissions. Lattice-based constructions, indeed, stand out for several advantages, as their reasonable parameter sizes and simple implementation make them versatile and efficient while ensuring strong security properties. They have at their core the *Learning With Errors* (LWE) problem. Since many LWE-based schemes require sampling from a discrete Gaussian distribution, we study methods for masking samplers and provide the first binomial sampler secure at high-order, which can be possibly adopted by several schemes submitted to the NIST standardization, like NEWHOPE [Alk+16], SABER [D'A+18], and KYBER [Bos+18]. Additionally, since lattice-based cryptography employs both Boolean masking and Arithmetic masking, we introduced new conversion algorithms from Boolean to Arithmetic masking, which improve the previous works and specifically fit the parameters employed in lattice-based constructions.

1.3. Organization of the thesis

The content of this thesis is organized as follows.

Preliminaries. Chapter 2 sets the necessary background for the comprehension of the thesis. It introduces different types of power analysis attacks, then moves to the description of diverse security models. It introduces the concepts of masking schemes and threshold probing model. Finally, it focuses on some cryptographic building blocks which are studied in the thesis.

Efficiency improvement in masking schemes. Chapter 3 presents the results related to the research conducted in [FPS17] at the scope of improving the randomness cost in Boolean masking. It proposes a strategy for reusing randomness among masked operations and it analyzes the application of such a method to the AES S-box.

Chapter 4 considers the Inner Product masking. It proposes improvements to the previous multiplication schemes in terms of randomness cost and discusses the potential of this masking scheme as an alternative to the more popular Boolean masking. The contribution of this chapter is part of [Bal+17].

Developing practical-oriented security models. Chapter 5 investigates the problem of formalizing security in the presence of physical defaults, proposing the so-called Robust probing model. In this context, it discusses the issues of providing composability guarantees while ensuring protection against glitches. The content of this chapter was published in [Fau+18].

Applying the masking countermeasure to lattice-based primitives. Chapter 6 treats the problem of protecting lattice-based primitives against power analysis attacks, as discussed in [Sch+19]. It proposes the first binomial sampler masked at generic security order and new efficient conversion algorithms from Boolean to Arithmetic masking for arbitrary modulo.

2. Background

This section aims at introducing the concept of provably security against side-channel attacks and providing the reader with the basic notions useful at the comprehension of this thesis.

We start by describing one of the most common types of side-channel attacks, i.e., the power analysis attack. We explain how such a practical attack is formalized in a leakage model and we present the major models proposed in the last 20 years. Then we introduce the concept of masking schemes, which constitute one of the most popular countermeasure against power analysis attacks, and we explain how to prove security of masked implementation within the framework of the threshold probing model. We end the chapter by recalling the main cryptographic building blocks used in this thesis.

2.1. Notation

In this dissertation, we denote a random sampling of an element x from a set X with $x \stackrel{\$}{\leftarrow} X$. Unless otherwise specified, any random sampling is from a uniform independent distribution. We use the bold font to define vectors, e.g., $\mathbf{x} = (x_1, \dots, x_n)$. The notation $\mathbf{x}|_X$ denotes $(x_i)_{i \in X}$ and the cardinality of a set X is $|X|$. The symbol \oplus indicates the binary sum, while the $+$ the arithmetic sum. The symbol \cdot is used to indicate both binary and arithmetic multiplication, according to the different context.

Notations relevant for specific chapters are defined in the respective chapters.

2.2. Recovering secrets: Power Analysis attacks

The first side-channel attack against a cryptographic implementation was described in the work of Kocher from 1996 [Koc96], where the author presented a technique which allows an adversary to recover the secret keys of famous public-key cryptosystems, such as Diffie-Hellman and RSA. A few years later, Kocher, Jaffe, and Jun presented in [KJJ99] the first power analysis attack, i.e., a side-channel attack which exploits the power consumption of the device, breaking the DES block cipher. From this moment on, power analysis attacks have grown in popularity and attracted the interest of the cryptographic community.

In order to perform a power analysis attack, an adversary makes use of two main components: a computer, connected to the targeted device, and an oscilloscope, which

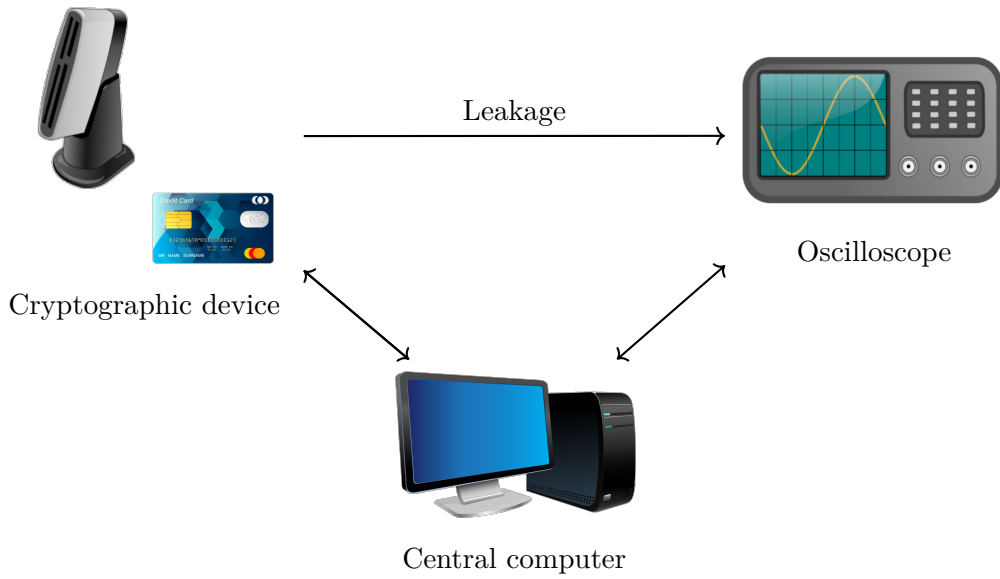


Figure 2.1.: Set-up of a power analysis attack

collects the measurements of the consumption in the form of traces. From the analysis of the patterns of such traces, the attacker obtains the information needed for recovering the secret key stored in the device. Figure 2.1 depicts the set-up just described when the targeted device is a smart card. Figure 2.2 represents an example of power consumption traces of the AES block cipher taken from the DPA attacks of [Bal+15] on an ARM Cortex-A8 processor running at 1 GHz. The trace presents some regularities, which correspond to the ten rounds of the AES-128¹.

Power analysis attacks can vary according to different parameters, like the strategy adopted, the requirements, the adversarial power, and they are generally classified into three categories: simple power analysis attacks, differential power analysis attacks, and profiled attacks.

2.2.1. Simple Power Analysis attack

In Simple Power Analysis (SPA in short) attacks, an adversary analyzes a single consumption trace, or multiple of them individually, over several cycle sequences [KJJ99]. The patterns observed can reveal information about the manipulated data and instructions, since the power consumption of the device, in general, is strictly related to the changes of the state. For instance, in symmetric cryptosystems the traces can show the

¹See Section 2.8.1 for more details on the AES-128 structure

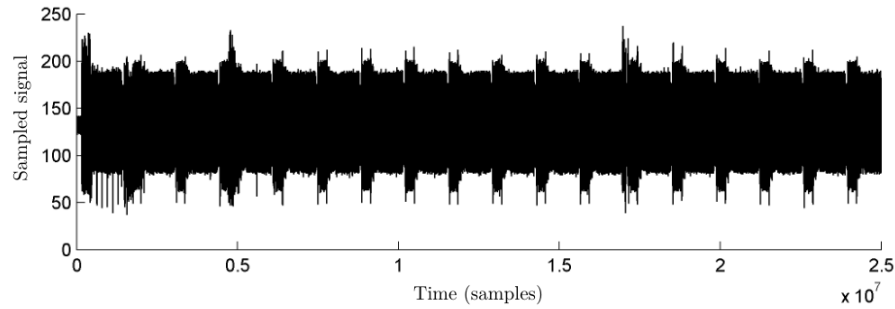


Figure 2.2.: Power consumption traces of an implementation of AES-128 [Bal+15]

number of rounds, as in Figure 2.2, the key length or sometimes memory accesses. In an extreme, but possible, scenario the traces can reveal the exact operand values, therefore if the computation depends on the secret key, the entire key-value can be successfully recovered. Assuming that the process of each bit affects the instantaneous dynamic power consumption, i.e., the power consumed while the operations are in process, higher power consumption peaks could, for instance, reveal a 1 bit and the lower ones could reveal a 0 bit.

A straightforward way to limit the effect of SPA attacks consists in avoiding dependencies between computation and secret key. As they are easy to counteract, the SPA attacks are less used in practice, compared to the Differential Power Analysis attack, described in the next section.

2.2.2. Differential Power Analysis attack

During a Differential Power Analysis (DPA in short) attack, an adversary first collects a number of power consumption traces corresponding to repeated executions of the targeted algorithm, using different inputs but the same fixed secret key and then analyzes the combination of them. Introduced in [KJJ99], we can nowadays distinguish between first-order and high-order attacks. In a first-order attack, an adversary exploits a single point of a power consumption trace, which corresponds to a targeted variable. In high-order attacks, instead, multiple points are exploited. In particular, we say that a DPA attack has order d when it combines the information of at most d points in the power traces. High-order DPA attacks are the most challenging to protect [Mes00].

The core idea of DPA attacks, as the authors explained in [KJJ99], is *key testing*. In this kind of strategy, the adversary performs DPA in order to confirm or reject hypothesis about an intermediate state of an implementation. In more detail, first, the attacker collects a set of N traces, as described above. Then he decides on a variable in the implementation to target, which depends on the input and a number of bits b of the

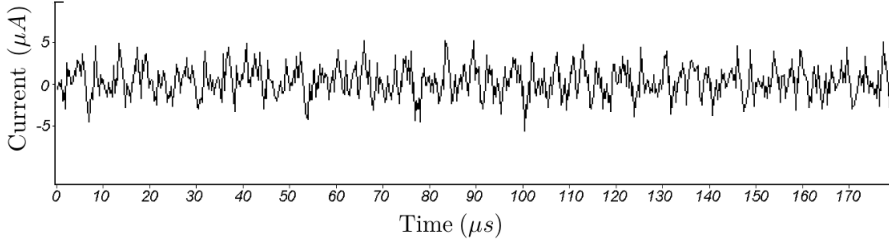


Figure 2.3.: Example of differential curve for the incorrect hypothesis of k_b [KJJ99]

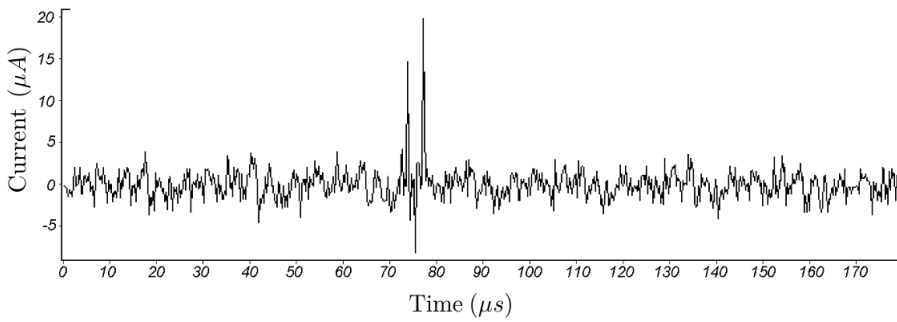


Figure 2.4.: Example of differential curve for the correct hypothesis of k_b [KJJ99]

key. We define this point $Y := f(X, k_b)$, with X being the input, f a function and k_b a b -bits portion of the key. All the possible 2^b b -bit-keys k_b^* are tested in the following way. First, they are used to compute the targeted value $Y_i := f(X_i, k_b^*)$, for all the different inputs X_i , with $i = 1, \dots, N$. Then, the plaintexts are divided into two groups for each key k_b^* and according to the value Y_i computed (for instance, if $LSB(Y_i) = 1$, its input X_i is in the first group, and if $LSB(Y_i) = 0$, then X_i is in the second group). At this point, the attacker computes the mean curve for both groups and calculates the difference between both means, so-called *differential curve*. The key hypothesis matching to the correct b -bits key corresponds to the highest difference of means, which appears on the differential curve as a peak. When the hypothesis of k_b^* is incorrect, indeed, the two groups created do not have significant differences, because the $LSB(Y_i)$ are random. Examples of differential curves with a correct and incorrect hypothesis of the key for the DES block cipher are depicted in Figures 2.3 and 2.4.

After the work of Kocher, Jaffe and Jun, new techniques have been developed in order to perform a DPA attack, in particular the *prediction model* has taken major interests. This consists in building a function of prediction, parametrized by the hypotheses of k^* , which predicts the power consumption. A distinguisher, i.e., a function which compares the predicted and real trace values, reveals the correct guess of the key. This corresponds to the value of k^* which shows the highest dependency between the real and predicted

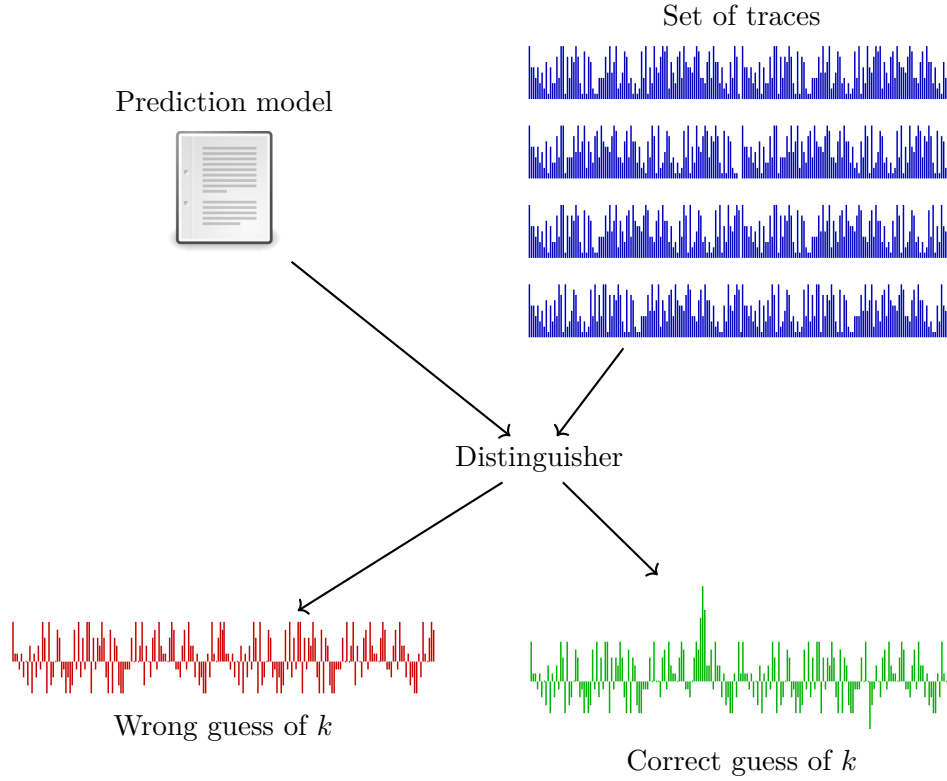


Figure 2.5.: DPA principle using a prediction model (Inspired by [Bel15])

traces. Several distinguishers are discussed in the literature, and a typical example is the Mutual Information Analysis (MIA), which we introduce below. A diagram of the DPA principle in modern applications is shown in Figure 2.5.

Mutual Information Analysis

The Mutual Information Analysis is a distinguisher introduced by Gierlichs, Batina, Tuyls, and Preneel in [Gie+08], which consists in measuring the dependence between two random variables, specifically the leakage at some point of interest and the prediction for each key hypothesis. In the case of continuous leakage \mathcal{L} and the discrete predictions \mathcal{P} , the mutual information is defined as:

$$MI(\mathcal{L}, \mathcal{P}) = \sum_{p \in \mathcal{P}} \int_{\mathbb{R}} \Pr[\mathcal{P} = p, \mathcal{L} = l] \log \frac{\Pr[\mathcal{P} = p, \mathcal{L} = l]}{\Pr[\mathcal{P} = p] \cdot \Pr[\mathcal{L} = l]}$$

The mutual information between two variables is always equal or greater than zero. Since it gives an estimation of the level of dependence between two random variables, it is equal

to zero in the case of a wrong hypothesis of the key. However, even a wrong prediction of the key can show dependencies with the leakage. The adversary exploits this property by selecting the key corresponding to the highest distinguisher value.

2.2.3. Profiling attacks

In profiling attacks the adversary improves the precision of SPA and DPA attacks, by employing techniques that involve a profiling phase. Among the various attacks in this category, the *template attacks* are the most powerful ones. Introduced by Chari, Rao and Rohatgi in [CRR03], they consist of collecting, during the initial profiling phase, a set of templates corresponding to known values of a targeted intermediate variable. A template is a collection of probability distributions that represents the aspect of power traces according to different keys used. This information helps to find detailed differences between power traces, and to make perfect key guesses for a single power trace. More precisely, after having recorded some leakage traces, the attacker makes use of statistical tests in order to perform a comparison of the traces against the template collection and to detect which value is most suitable to be the correct subkey.

2.3. Leakage models

The task of designing a security model that realistically captures the effect of side-channel attacks is not easy to achieve, and in the last 20 years many leakage models have been proposed. We use in the following the classification from [KR19], which distinguishes between *memory leakage models* and *computation leakage models*. In the former ones, the adversary is able to receive an arbitrary bounded-length leakage on the secret key, which is polynomial-time computable. In the latter ones, the leakage also comes from the intermediate values, computed during the execution of the cryptographic algorithm. On top of these basic distinctions, there are many other important differences. For instance, the two models usually have a different level of restrictions on the leakage provided to the attacker. On the one hand, in memory leakage models, the leakage does not have many constraints, and we generally represent it as an arbitrary bounded function of the whole secret. On the other hand, computation leakage models consider a higher level of restrictions. They are often based on assumptions, such as that whenever parts of the computation are separated in space or time, the respective leakage functions are independent from each other, or sometimes that some memory is leak-free. Moreover, the computation leakage models take often into account continuous leakage over several uses of the secret key, which results in the fundamental need to update the secret memory in order not to break security in this model. On the contrary, the memory leakage models do not present this problem, unless a few exceptions, because they mainly consider one-time leakage.

In the field of memory leakage models, the first influential works are from Dziembowski [Dzi06], and Di Crescenzo, Lipton and Walfish [DLW06], who introduced the so-called *bounded retrieval model*, which assumes that there is a natural bound on the overall amount of information the attacker can retrieve, and, based on such an assumption, it expresses the leakage as an arbitrary and independent parameter of the system. The adversary has access to a polynomial-time computable leakage function $f(K)$, with K being the secret key, with bounded output size. A common protection against such an attack consists in making the secret key large, at the cost of increasing the running times by a factor at most polylogarithmic in K . Afterwards, in [AGV09] the authors defined the *bounded memory leakage model*, which was successively generalized in [DKL09] to the so-called *auxiliary input leakage model* and to the continual setting in [Bra+10; Dod+10]. This line of works defines the leakage not anymore as a parameter but as a function of the secret key length. This thesis has not the intention of deepening more the theories on memory leakage models. It instead focuses more on the computation leakage models.

The scientific literature abounds of several computation leakage models, differing mostly for their levels of realism and ease of usage in security proofs. One of the first models introduced has been the *t-threshold probing model*, which Ishai *et al.* presented in their seminal work [ISW03]. The probing model interprets the physical leakage captured by the power analysis attack as a set of intermediate values of the algorithm. More precisely, an adversary has access to a bounded number of internal variables, i.e., wires of the implementation, which are more commonly called *probes*. A circuit is defined to be secure in this model if any subset of at most t probes do not provide enough information to recover the sensitive data of the computation. This model has the drawback of being ideal, as it does not fully catch the reality of the physical leakages, which usually reveal information related to the whole implementation, and not only on a smaller set of variables. However, as we will discuss later, the simplicity of the probing model allows the construction of efficient compilers that transform any implementation into a private circuit, i.e., a circuit resilient against a probing attack, with a blow-up of the circuit size by a factor $\mathcal{O}(t^2)$.

Micali and Reyzin in [MR04] made a step in the direction of more realistic leakage models. The authors proposed the principle that *Only Computation Leaks Information*, and considered that the leakage takes into account calculations, and not data. They assumed, therefore, that an implementation can still hide some secrets. More precisely, an adversary can make a measurement on any computation step containing parts of sensitive data, like secret keys, internally generated randomness, and outcomes of previous computations performed on cryptographic keys. However, no adversary can measure data that is not involved in a computation step at a particular point of time.

Later in 2013, Prouff and Rivain introduced a new leakage model which captures well the power analysis attacks. The model was presented in [PR13], based on the work

of [Cha+99], and it is called δ -noisy leakage model. It expresses the leakage of an elementary operation on an input x in terms of a *noisy leakage function* $f(x)$. More precisely, during an attack in this setting, the adversary has access to computational values, sampled according to a given distribution, say X . The leakage function f is a randomized function such that the distance (measured with the euclidean norm) between the distribution of the values X and the distribution of these values, given their leakage function $X|f(X)$, is bounded by a fixed value δ . In the paper, the authors additionally showed the first security proof for the evaluation of a whole block cipher. The security strongly relies on the contribution of [Cha+99], where the authors proved that employing additive secret sharing on the variables of an implementation reduces drastically the information leaked from the sensitive data. The proof is possible thanks to the strong assumption that leak-free gates exist, i.e., that part of the computation does not leak any information. These leak-free gates are used in order to refresh internal values. The main limitation of this model is the difficulty in designing security proofs.

The latter drawback was mitigated by the work of Duc, Dziembowski and Faust [DDF14], who presented a reduction between the noisy leakage model and the threshold probing one and overcome the requirement of employing leakage-free gates. In the paper, the authors recalled the so-called *random probing model*, already introduced in [ISW03], where every intermediate value is leaked to the adversary with probability ϵ . Such a model is used in order to build an intermediate step in the reduction, i.e., first the authors show a reduction from the δ -noisy leakage model to the random probing model, and then from the latter one to the probing model. In particular, by using the statistical distance in place of the euclidean norm in the definition of noise, they showed that proving a cryptographic implementation to be secure in the t -threshold probing model implies its security in the δ -noisy leakage model as well. Specifically, given a set \mathcal{X}^l of intermediate variables, proving the security in the δ -noisy leakage model for a set $(x_1, \dots, x_l) \in \mathcal{X}^l$ can be achieved by proving the security in the t -probing model, with $t = 2\delta(l-1)|\mathcal{X}|$. This bound constitutes the main drawback of the reduction. In [DFS15b] the bound was later improved, but the new proof requires the adoption of leak-free components again. In the paper, the authors introduced a new leakage model, the *average probing model*, which is a generalization of the random probing model. They proved a tight equivalence between the noisy leakage model and the average probing model and show that the compiler from [ISW03] is secure in the average probing model, assuming a leak-free component. Such kinds of bounds are often the consequence of strategies deployed in the proofs for theoretical reasons, and they do not reflect reality, as it has been later observed in the work of Duc, Faust and Standaert [DFS15a].

Balasch *et al.* pointed out in [Bal+14] that other kinds of physical effects can influence the leakage, e.g., the so-called *transitions* [Bal+14; Cor+12] and *glitches* [MPG05; MPO05]. The former ones typically happen in software implementations and consist in

memory recombinations, which happen when a register is used consecutively to store two different variables. In this case, an adversary can observe both the first and the second value stored. The glitches, instead, are combinatorial recombinations, which can be encountered in hardware implementations, when differences in the timing signals produce undesired transitions at the output of logical gates, before the gate switches to the final output.

Barthe *et al.* in [Bar+17] introduced a new model, the *bounded moment model*, that formalizes a weaker notion of security order, in line with the intuitions of the practical side-channel community. The authors observed that the attacker’s intent to determine higher-order statistical moments is exponentially difficult in the number of shares if their leakages are independent of the sensitive data and sufficiently noisy. Therefore they related the security order in the bounded moment model to the lowest key-dependent statistical moment of the leakage distribution.

The rest of this thesis work will mainly consider in the t -threshold probing model, and we will investigate the physical defaults described above as well.

2.4. Masking, a countermeasure against SCA

Since the discovery of side-channel attacks, several techniques to defeat them have been designed. These strategies have in common that they aim at reducing, and often eliminating, the dependencies between the physical leakage of a cryptographic implementation and the sensitive values produced during the computations of the algorithm, i.e., those values which, if observed through a side-channel attack, could otherwise leak secret data.

The various countermeasures aim at preserving the black-box security properties of the cryptosystem and mainly differ from each other for the level they target. Some of them act at the leakage level, trying to randomize the characteristics of the power consumption or to make it uniform through the execution of the algorithm. To this end, techniques as [CK10; Vey+12] inject changes in the time dimension of the leakage, by inserting dummy operations or by altering the sequence of the computation. Other strategies, as [KJJ01; Sha00], act on the amplitude domain of the leakage by increasing the noise in a signal or by lowering the signal itself.

Other types of side-channel countermeasures target the protocol level by reducing the frequency of cryptographic operations performed using the same key [KJJ99]. For instance, a common technique is the one of re-keying [Med+10], which consists in changing the secret frequently.

Masking schemes target at the algorithmic level and aim at randomizing the intermediate computation of the cryptographic device. Goubin and Patarin in [GP99] and Chari *et al.* in [Cha+99] introduced such a strategy in 1999 and it is one of the most important techniques deployed in order to counteract side-channel attacks and the main topic of

this thesis.

The countermeasure works by applying the principle of secret sharing to the sensitive variables of a cryptographic implementation, for instance, intermediate values of block ciphers depending on a known plaintext and a key. More formally, given a chosen operation \diamond and a DPA attack order d , masking a sensitive variable x means to encode it, by splitting it into a number $n = d + 1$ of *shares* x_1, \dots, x_n such that the combination of them according to \diamond gives the original variable x , i.e., $x = x_1 \diamond \dots \diamond x_n$, and any combination of less than n shares does not reveal any sensitive information. Generally, $n - 1$ of the shares are chosen uniformly at random, and the n^{th} one is chosen such that the secret can be recombined.

As Chari *et al.* showed in [Cha+99], under the fundamental assumption that each leaked sample depends on a bounded number of shares and such leakage is sufficiently noisy, the difficulty of performing a successful DPA attack of order d increases exponentially with the number of shares n , also called the *masking order*.

Different types of masking schemes exist according to the different encoding functions, i.e., to the operation \diamond and the field the variables lie in. The choice of a certain masking scheme depends on the algorithm to mask. For instance, when an algorithm mostly performs arithmetic operations, it is convenient to use *Arithmetic masking*, i.e., to consider \diamond as a modular addition in the arithmetic field \mathbb{F}_{2^k} . On the other hand, when most of the operations are boolean it is advantageous to consider \diamond as the addition or multiplication, in the field \mathbb{F}_2 . The most common masking is the *Boolean masking*, i.e., $\diamond = \oplus$. In this case, the complexity of the masked linear operations is only $\mathcal{O}(n)$, as they simply operate component-wise, while masking the non-linear operations is more expensive, and it requires a complexity of $\mathcal{O}(n^2)$.

When a cryptographic algorithm uses both arithmetic and boolean variables, the masking scheme is chosen according to the different cases, and a conversion algorithm allows to pass from one kind of masking scheme to the other.

Besides the aforementioned Boolean and Arithmetic masking, other examples of masking schemes are the Affine masking [Fum+10], the Polynomial masking [GM11; PR11] and the Inner Product masking [Bal+12; BFG15; DF12; GR12]. They differ from each other not only by their construction but also by their efficiency and security guarantees. As an example, compared to the Boolean masking, the Inner Product masking shows less evidence of leakage in practice but provides lower performance.

2.5. Conversion algorithms between Arithmetic and Boolean masking

We mentioned previously the existence of conversion algorithms, which allow transforming Boolean masking into Arithmetic ones, and vice versa. Goubin in [Gou01] proposed for

the first time a transformation from Boolean to Arithmetic masking (B2A) based on the fact that the function $\Phi(x, r) : \mathbb{F}_{2^k} \times \mathbb{F}_{2^k} \mapsto \mathbb{F}_{2^k}$ such that

$$\Phi(x, r) = (x \oplus r) - r \pmod{2^k} \quad (2.1)$$

is affine in r over \mathbb{F}_2 , i.e., $\Phi(x, r' \oplus r'') = \Phi(x, r') \oplus \Phi(x, r'')$. The algorithm has a run time complexity of $\mathcal{O}(1)$, i.e., independent of the size of the inputs k , and it is, therefore, very efficient. However, the authors provided a security proof only against a first-order adversary.

The first B2A algorithm secure at higher orders was presented much later, by Coron *et al.* in [CGV14]. The new algorithm relies on a different approach than the previous work. As a first step, it initializes $n - 1$ shares $(A_i)_{1 \leq i \leq n-1}$ with random samples in \mathbb{F}_{2^k} , and uses them to generate a random encoding \mathbf{A}' of the form $\sum_{i=1}^n A'_i = -\sum_{i=1}^{n-1} A_i \pmod{2^k}$. Successively, a higher-order secure Arithmetic-to-Boolean (A2B) conversion algorithm takes as input such encodings and outputs the Boolean shares $\bigoplus_i y_i = \sum_i A'_i \pmod{2^k}$, which are then added to the input encoding \mathbf{x} , leading to:

$$\bigoplus_i z_i = \bigoplus_i x_i + \bigoplus_i y_i = x - \sum_{i=1}^{n-1} A_i \pmod{2^k}.$$

By using a function which securely decodes a given input encoding, the remaining share of \mathbf{A} is set to the decoding of \mathbf{z} . As a result:

$$\sum_{i=1}^n A_i = \sum_{i=1}^{n-1} A_i + (x - \sum_{i=1}^{n-1} A_i) = x \pmod{2^k}.$$

Later in [BCZ18], the affine relation given in Equation (2.1) inspired another higher-order B2A conversion algorithm, which follows previous work of [HT19] and [Cor17].

This conversion is not based on the Boolean-masked addition of shares, therefore it has run time independent of the input bit size, and it is, therefore, particularly efficient for small values of n . However, for an increasing number of shares, the performance drops, showing the asymptotic run time complexity of $\mathcal{O}(2^n)$.

In the context of Arithmetic-to-Boolean (A2B) conversion algorithms, there is only one secure higher-order A2B algorithm to the best of our knowledge, published in [CGV14]. Its underlying concept is rather simple. The algorithm works by, first transforming each share of the arithmetic input encoding \mathbf{A} into a Boolean encoding with n shares. Then, such n Boolean encodings are added together using a Boolean-masked addition algorithm, producing a Boolean encoding \mathbf{x} with $\bigoplus_i x_i = \sum_i A_i \pmod{2^k}$. This basic version has a cubic complexity of $\mathcal{O}(n^3 \cdot k)$. In the same paper, the authors additionally propose a strategy to improve the conversion, resulting in a quadratic complexity of $\mathcal{O}(n^2 \cdot k)$. The core idea is adopt a recursive method to sum the input shares A_i , instead of summing them up one by one. More precisely, the improved scheme splits the n shares into two

halves of $\lfloor \frac{n}{2} \rfloor$ and $\lceil \frac{n}{2} \rceil$, and recursively calls itself for the two halves. Finally, it adds together the resulting encodings:

$$\begin{aligned} x &= (A_1 + \dots + A_{\lfloor n/2 \rfloor}) + (A_{\lfloor n/2 \rfloor + 1} + \dots + A_n) \\ &= (y_1 \oplus \dots \oplus y_{\lfloor n/2 \rfloor}) + (z_1 \oplus \dots \oplus z_{\lceil n/2 \rceil}) \\ x &= x_1 \oplus \dots \oplus x_n. \end{aligned}$$

In the next section, we will see more details on the construction of masked algorithms, and we will show why they are secure against side-channel attacks.

2.6. Proving security in the probing model

The most important and convenient model to analyze masking schemes is the t -threshold probing model. For this reason, and for being widely deployed in the field of leakage resilient cryptography, among all the leakage models we focused on the probing model.

In this context, we represent a cryptographic implementation as a *circuit*. We call *private circuit* a transformation of a circuit into another one, which is secure in the probing model. According to the description in [ISW03], circuits can be distinguished into *deterministic* and *randomized* ones. A *deterministic circuit* C is represented as a direct acyclic graph whose vertexes are boolean gates and whose edges are wires. A *randomized circuit* is a circuit augmented with random-bit gates. A random-bit gate is a gate with fan-in 0 that produces a random bit and sends it along its output wire; the bit is selected uniformly and independently. As pointed out in [Ish+13], a t -*private circuit* is a randomized circuit which transforms a randomly encoded input into a randomly encoded output while providing the guarantee that the joint values of any t wires reveal nothing about the input. More formally a *private circuit* is defined as follows.

Definition 2.1 (Private circuit [Ish+13]). *A private circuit for $f : \mathbb{F}_2^{m_i} \rightarrow \mathbb{F}_2^{m_o}$, with $m_i, m_o \in \mathbb{N}$, is defined by a triple (I, C, O) , where*

- $I : \mathbb{F}_2^{m_i} \rightarrow \mathbb{F}_2^{\hat{m}_i}$ is a randomized input encoder;
- C is a randomized boolean circuit with input in $\mathbb{F}_2^{\hat{m}_i}$, output in $\mathbb{F}_2^{\hat{m}_o}$ and uniform randomness $r \in \mathbb{F}_2^n$
- $O : \mathbb{F}_2^{\hat{m}_o} \rightarrow \mathbb{F}_2^{m_o}$ is an output decoder

C is said to be a t -*private implementation* of f with encoder I and decoder O if the following requirements hold:

- *Correctness*: For any input $w \in \mathbb{F}_2^{m_i}$ we have $\Pr[O(C(I(w), \rho)) = f(w)] = 1$, where the probability is over the randomness of I and ρ ;

- *Privacy*: For any $w, w' \in \mathbb{F}_2^{m_i}$ and any set \mathcal{P} of t wires (also called *probes*) in C , the distributions $C_{\mathcal{P}}(I(w), \rho)$ and $C_{\mathcal{P}}(I(w'), \rho)$ are identical, where $C_{\mathcal{P}}$ denotes the set of t values on the wires from \mathcal{P} (also called *intermediate values*).

The goal of a *t-limited attacker*, i.e., an attacker who can probe at most t wires, is to find a set of probes \mathcal{P} and two values $w, w' \in \mathbb{F}_2^{m_i}$ such that the distributions $C_{\mathcal{P}}(I(w), \rho)$ and $C_{\mathcal{P}}(I(w'), \rho)$ are not the same.

For simplifying the notation, in Definition 2.1 we considered the easiest case, when the underlining field is the binary field, and consequently the wires carry bits. However, the definition is still valid in larger fields such as \mathbb{F}_{2^n} . In the first case we consider the Boolean masking, while in the second case the Arithmetic one. The encoder I consist in the encoding function specified by the masking scheme, and the output decoder O is the procedure which reconstructs the sensitive variable encoded. As an example, in the case of Boolean masking, the input encoder I maps every input value $x \in \mathbb{F}_2$ into n binary shares (r_1, \dots, r_n) , where the first $n - 1$ values are chosen at random and $r_n = x \oplus r_1 \oplus \dots \oplus r_{n-1}$. On the other hand, the output decoder O takes the n bits y_1, \dots, y_n produced by the circuit and decodes the values in $y = y_1 \oplus \dots \oplus y_n$.

In its internal working a private circuit is composed by *gadgets*, namely transformed gates which perform functions taking as input a set of masked inputs and outputting a set of masked outputs. In particular, we distinguish between linear operations (e.g., addition), which can be performed by applying the operation to each share separately, and non-linear functions (e.g., multiplication), which process all the shares together and make use of additional random bits. A particular case of randomized gadget is the *refreshing* gadget, which takes as input the sharing of a value x and outputs randomized sharing of the same x .

One can show the security of a circuit in the probing model by proving its *perfect simulatability* property, i.e., that any set of at most t intermediate values is jointly independent of the sensitive values. In practice, it is sufficient to show that any set of potential adversarial observations can be perfectly simulated by at most t shares of the inputs. The simulation method, despite perfectly valid, is costly and can easily lead to mistakes difficult to discover, as the proof designer needs to list all the possible subsets of intermediate values and show for each case their simulation. The complexity and length of the proof naturally increases with the security level and the circuit size. In order to overcome the latter problem, the definition of probing security has been strengthened in [Bar+15a] such that it could give a further guarantee: composability. With the introduction of the notions of t -Non Interference (t -NI) and t -Strong Non Interference (t -SNI), defined below, proving the probing security of a circuit can be done modularly, by first showing with a simulation-based proof the security of the individual gadgets, and then exploiting the properties of t -NI and t -SNI to show that the composition of the

gadgets is still secure. The following definitions and lemma from [Bar+15a] formalize the notion of *t-Non Interference* and show that this is also equivalent to the concept of *simulatability*.

Definition 2.2 ((\mathcal{S}, Ω)-Simulatability, (\mathcal{S}, Ω)-Non Interference). *Let \mathbf{g} be a gadget having m inputs $(a^{(1)}, \dots, a^{(m)})$ each split in n shares and Ω be a set of t adversary's observations. Let $\mathcal{S} = (\mathcal{S}_1, \dots, \mathcal{S}_m)$ such that $\mathcal{S}_i \subseteq \{1, \dots, n\}$ and $|\mathcal{S}_i| \leq t$ for all i .*

1. *The gadget \mathbf{g} is (\mathcal{S}, Ω)-simulatable (or (\mathcal{S}, Ω) – SIM) if there exists a simulator which, by using only $(a^{(1)}, \dots, a^{(m)})|_{\mathcal{S}} := (a_{|\mathcal{S}_1}|^{(1)}, \dots, a_{|\mathcal{S}_m}|^{(m)})$, can simulate the adversary's view, where $a_{|\mathcal{S}_j}|^{(k)} := (a_i^{(k)})_{i \in \mathcal{S}_j}$.*
2. *The gadget \mathbf{g} is (\mathcal{S}, Ω)-Non Interfering (or (\mathcal{S}, Ω) – NI) if, for any pair $\mathbf{s}_0, \mathbf{s}_1 \in (\mathbb{F}_2^m)^n$ such that $\mathbf{s}_0|_{\mathcal{S}} = \mathbf{s}_1|_{\mathcal{S}}$, the adversary's views of \mathbf{g} respectively on input \mathbf{s}_0 and \mathbf{s}_1 are identical, i.e., $\mathbf{g}(\mathbf{s}_0)|_{\Omega} = \mathbf{g}(\mathbf{s}_1)|_{\Omega}$.*

Often when we talk about *simulatability* of a gadget we implicitly mean that for every observation set Ω with $|\Omega| \leq t$, where t is the security order, there exists a set \mathcal{S} as in Definition 2.2 such that the gadget is (\mathcal{S}, Ω) – SIM.

Lemma 2.3. *Let \mathbf{g} be a gadget with m inputs and let $\mathcal{S} = (\mathcal{S}_1, \dots, \mathcal{S}_m)$, where, for each $i = 1, \dots, m$, $|\mathcal{S}_i| \leq t$ and $\mathcal{S}_i \subseteq \{1, \dots, n\}$. Let Ω be an observation set, with $|\Omega| \leq t$. The gadget \mathbf{g} is (\mathcal{S}, Ω) – SIM if and only if \mathbf{g} is (\mathcal{S}, Ω) – NI, with respect to the same sets (\mathcal{S}, Ω) .*

Definition 2.4 (*t*-Non-Interference). *A gadget \mathbf{g} is *t*-Non-Interfering (*t* – NI) if and only if for every observation set Ω , with $|\Omega| \leq t$, there exists a set \mathcal{S} , with $|\mathcal{S}| \leq t$, such that \mathbf{g} is (\mathcal{S}, Ω) – NI.*

When applied to composed circuits, the definition of *t* – NI is not enough to guarantee the privacy of the entire circuit. While the notion of *t* – NI is not sufficient to argue about secure composition of gadgets, the stronger definition of *t* – Strong Non-Interference (*t* – SNI), introduced in [Bar+15a] as well, allows a secure composition of gadgets.

Definition 2.5 (*t* – Strong Non-Interference). *An algorithm \mathcal{A} is *t* – Strong Non-Interferent (*t* – SNI) if and only if for any set of t_1 probes on intermediate values and every set of t_2 probes on output shares with $t_1 + t_2 \leq t$, the totality of the probes can be simulated by only t_1 shares of each input.*

Algorithm 2.1 Addition algorithm with $n \geq 2$ shares

Input: Shares $(a_i)_{1 \leq i \leq n}$ and $(b_i)_{1 \leq i \leq n}$ in \mathbb{F}_{2^k} , such that $\sum_i a_i = a$ and $\sum_i b_i = b$ **Output:** Shares $(c_i)_{1 \leq i \leq n}$ in \mathbb{F}_{2^k} , such that $\sum_i c_i = a + b$ 1: **for** $i = 1$ to n **do**2: $c_i = a_i + b_i$;

Algorithm 2.2 ISW multiplication algorithm with $n \geq 2$ shares

Input: Shares $(a_i)_{1 \leq i \leq n}$ and $(b_i)_{1 \leq i \leq n}$ in \mathbb{F}_{2^k} , such that $\sum_i a_i = a$ and $\sum_i b_i = b$ **Output:** Shares $(c_i)_{1 \leq i \leq n}$ in \mathbb{F}_{2^k} , such that $\sum_i c_i = a \cdot b$ 1: **for** $i = 1$ to n **do**2: **for** $j = i + 1$ to n **do**3: $r_{i,j} \xleftarrow{\$} \mathbb{F}_{2^k}$;4: $r_{j,i} \leftarrow (r_{i,j} + a_i \cdot b_j) \oplus a_j \cdot b_i$;5: **for** $i = 1$ to n **do**6: $c_i \leftarrow a_i \cdot b_i + \sum_{j=1, j \neq i}^n r_{i,j}$;

Informally, it means that the simulator can simulate the adversary's view, using a number of shares of the inputs that is independent from the number of probed output wires.

Building gadgets secure in these models can be a more or less difficult task according to the kind of operation to mask and the kind of masking. As mentioned previously, linear operations, such as the addition, are trivially secure in the probing model by applying the relative function component-wise, as depicted in Algorithm 2.1, which is $t - \text{NI}$. We point out that these functions cannot be proven to be $t - \text{SNI}$, unless some randomness is injected in the outputs.

On the other hand, multiplication schemes are more complex to mask and they need to employ some randomness. An example of $t - \text{SNI}$ multiplication algorithm is the famous ISW scheme in Algorithm 2.2, introduced in [ISW03] and proven to be $t - \text{SNI}$ in [Bar+15a]. A $t - \text{SNI}$ refreshing scheme is Algorithm 2.3, introduced in [DDF14] by Duc *et al.* and proven to be $t - \text{SNI}$ by Barthe *et al.* in [Bar+15a].

As pointed out in [RP10] and [Cor+13], secure multiplication schemes, like ISW, require that the two masks in input are mutually *independent*. This condition is satisfied in two cases: when at least one of the two inputs is taken uniformly at random or when at least one of the two inputs is refreshed by means of a secure refreshing using completely fresh and independent randomness, as in Algorithm 2.3.

Algorithm 2.3 Multiplicative refreshing scheme

Input: Shares $(a_i)_{1 \leq i \leq n}$ in \mathbb{F}_{2^k} , such that $\sum_i a_i = a$; random shares $(r_{ij})_{1 \leq i \leq n, i+1 \leq j \leq n}$

Output: Shares $(c_i)_{1 \leq i \leq n}$ in \mathbb{F}_{2^k} , such that $\sum_i c_i = a$

```

1: for  $i = 1$  to  $n$  do
2:    $c_i = a_i$ ;
3: for  $i = 1$  to  $n$  do
4:   for  $j = i + 1$  to  $n$  do
5:      $c_i = c_i + r_{i,j}$ 
6:      $c_j = c_j - r_{i,j}$ 

```

2.7. Threshold Implementation

Threshold Implementation (TI) schemes are another kind of countermeasures against DPA attacks. They are based on masking schemes and, compared to the probing model, they have the additional advantage to take into account physical effects such as glitches.

When dealing with such additional physical defaults, we require circuits to be stateful. For this purpose, the circuit model presented in the previous section is augmented with *memory gates* which, on every invocation of the circuit, output the previous input to the gate and store the current input for the next invocation. Such abstractions can be reasonably and efficiently instantiated in practice, using true- or pseudo random number generators for the random gates and registers (synchronized by a clock signal) for the memory gates.

In order to implement a boolean function $f : \mathbb{F}_2^{m_i} \rightarrow \mathbb{F}_2^{m_o}$, every input value x has to be split into n shares (x_1, \dots, x_n) such that $x = x_1 \oplus \dots \oplus x_n$, using the same procedure seen in private circuits. We denote with \mathbf{C} the output distribution $f(\mathbf{X})$, where \mathbf{X} is the distribution of the encoding of an input x . The function f is then shared in a vector of functions f_1, \dots, f_n , called component functions, which must satisfy the following properties:

1. **Correctness:** $f(x) = \sum_{i=1}^n f_i(x_1, \dots, x_n)$
2. **t -Non-Completeness:** any combination of up to t component functions f_i of f must be independent of at least one input share x_i .
3. **Uniformity:** the probability $\Pr(\mathbf{C} = \mathbf{c} | \mathbf{c} = \sum_{i=1}^n c_i)$ is a fixed constant for every \mathbf{c} , where \mathbf{c} denotes the vector of the output shares.

The last property requires that the distribution of the output is always a random sharing of the output, and can be easily satisfied by refreshing the output shares. The non-completeness property, instead, is used to prevent that any combinatorial logic has access to all the shares of an encoded sensitive variable.

TI schemes are strongly related to private circuits. First, they solve a similar problem

Algorithm 2.4 TI multiplication algorithm for $t = 1$ **Input:** Shares $(a_i)_{1 \leq i \leq 3}$ and $(b_i)_{1 \leq i \leq 3}$ in \mathbb{F}_2 , such that $\bigoplus_i a_i = a$ and $\bigoplus_i b_i = b$ **Output:** Shares $(c_i)_{1 \leq i \leq 3}$ in \mathbb{F}_2 , such that $\bigoplus_i c_i = a \cdot b$ 1: $c_1 \leftarrow a_2 b_2 \oplus a_1 b_2 \oplus a_2 b_1$;2: $c_2 \leftarrow a_3 b_3 \oplus a_2 b_3 \oplus a_3 b_2$;3: $c_3 \leftarrow a_1 b_1 \oplus a_3 b_1 \oplus a_1 b_3$;

of formalizing privacy against a t -limited attacker and moreover, as shown in [Rep+15], the TI algorithm for multiplication is equivalent to the scheme proposed by Ishai *et al.* in [ISW03]. An example of multiplication scheme at first order of security is depicted in Algorithm 2.4

We additionally point out that the aforementioned properties of TI imply simulatability of the circuit. Indeed, if a function f satisfies properties 1 and 2, then an adversary who probes t or fewer wires will get information from all but at least one input share. Therefore, the gadget \mathbf{g} implementing such a function is $t - \mathbf{NI}$ and due to Lemma 2.3 is simulatable. As a consequence, proving security in the TI model does not require to provide a simulation of the gadgets, but simply requires to show that the properties of correctness, non-completeness and uniformity are satisfied.

Unfortunately, such a simulatability property, at least in the original definition of TI, is not proven to be guaranteed also when algorithms compose. However, even though the concept of composability is not explicitly analyzed in TI, differently than in the probing model, some works show that first-order TI benefits sometimes from a sort of composability which can allow to protect full block cipher executions with a minimum amount of randomness [Pos+11]. The same does not hold for higher-order TIs. Indeed the first try to construct a higher-order TI in [Bil+14] failed due to a composability flaw [Rep15; Rep+15].

2.8. Cryptographic building blocks

In this section, we introduce the main cryptographic building blocks that are considered in this thesis. In the context of symmetric cryptography, we analyze the popular AES block cipher, while when treating lattice-based cryptography we consider the NEWHOPE key exchange.

2.8.1. A symmetric key encryption scheme: the AES

In modern cryptography, thanks to their efficiency, the symmetric algorithms are generally preferred to the asymmetric ones. In this setting, a widely adopted family of algorithms that ensure secure communication is represented by the encryption schemes. Among

them, the block cipher algorithms are the most popular.

In this thesis, we consider the AES (Advanced Encryption Standard), the most commonly employed block cipher algorithm. Joan Daemen and Vincent Rijmen initially proposed the scheme under the name of *Rijndael* and the NIST chose it in 2001 [RD01] to become the new encryption standard, after a five-years standardization process.

AES algorithm consists of modifying a 4×4 array of bytes, called the *state*, in a series of rounds. Like in many block ciphers, such rounds consist of a key addition, some linear transformations, and a non-linear transformation. In the initialization, the state is set equal to the input of the cipher. Then, the rounds follow, each executing the procedures described below:

1. **AddRoundKey:** A 128-bit sub-key is obtained from the master key and XORed with the state. The state array is updated accordingly.
2. **SubBytes:** At this stage, a lookup table, commonly called *S-box*, is used to represent a bijection over $\{0,1\}^8$. According to such a substitution table, each byte of the state array is replaced by another byte. The S-box constitutes the non-linear transformation of the AES.
3. **ShiftRows:** We cyclically shift $i - 1$ places to the left the bytes in each row i of the state array, where each row is enumerated from 1 to 4.
4. **MixColumns:** We apply an invertible transformation to the four bytes in each column. According to such a transformation, any two inputs differing each other in $b > 0$ bytes correspond to two outputs differing in at least $5 - b$ bytes.

In the final round, in place of MixColumns we have once more AddRoundKey. The AES block cipher has a 128-bit block length and can employ 128-, 192-, or 256-bit keys. The key length does not affect the structure of each round, but only the key schedule and the number of rounds, which can vary among 10, 12, or 14.

The main challenge in protecting such a block cipher against power analysis attackers lies in masking the non-linear transformation, which is represented by the S-box. The AES S-box **SubBytes** is the right-composition of an affine transformation over \mathbb{F}_2^8 with the power function $x \mapsto x^{254}$ over the field $\mathbb{F}_2^8 \equiv \mathbb{F}_2[x]/(x^8 + x^4 + x^3 + x + 1)$. The affine transformation is usually straightforward to mask. Therefore the main focus of research is on masking the power function. In this thesis we consider Rivain and Prouff's algorithms from [Cor+13; RP10], depicted in Figure 2.6, where **Mult** is a multiplication scheme and \mathcal{R} a refreshing scheme. In order to guarantee a $t - \text{SNI}$ security of the algorithm, the refreshing schemes and the third multiplication scheme have to be $t - \text{SNI}$, and the rest of the multiplication schemes $t - \text{NI}$.

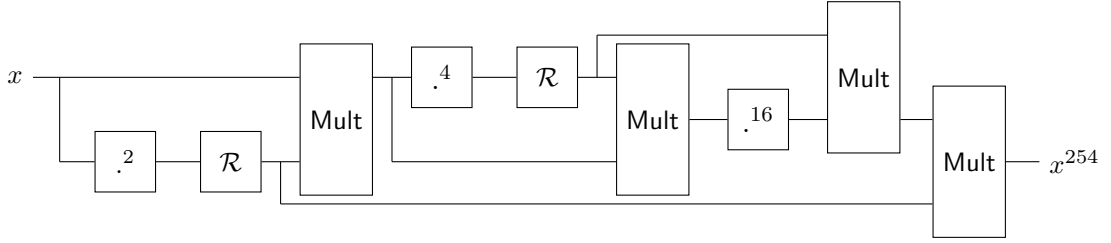


Figure 2.6.: Inversion chain of the AES S-box from [Cor+13; RP10]

2.8.2. A lattice-based key exchange protocol: the NewHope

We now briefly describe a construction, the NEWHOPE protocol, in the field of lattice-based cryptography, which we use in this thesis for evaluating the applicability of our contributions in the area. Before introducing it, we provide some foundations to post-quantum cryptography.

Learning With Error

As it is the case for modern cryptography, the security of a cryptographic scheme against a quantum adversary relies as well on the hardness of a computational problem. Many primitives in lattice-based cryptography base their security properties on the hardness of the Learning With Error (LWE) problem [Reg05], or one of its variants. LWE is believed to be hard, based on certain assumptions regarding the worst-case hardness of standard lattice problems, and it is defined as follows.

Definition 2.6 (Learning With Errors problem). *Let n, m, q be positive integers and χ be a distribution over \mathbb{Z} . Given $\mathbf{s} \leftarrow_{\chi} \mathbb{Z}^n$ let $\mathcal{D}_{\mathbf{s}, \chi}$ be the distribution that samples $\mathbf{a} \leftarrow_{\$} \mathbb{Z}_q^n$ and $e \leftarrow_{\chi} \mathbb{Z}$ and then returns $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$.*

- *The search LWE problem is to find \mathbf{s} , given $n, q \geq 2$, and m independent samples from $\mathcal{D}_{\mathbf{s}, \chi}$.*
- *The decisional LWE problem is to distinguish whether $(\mathbf{a}_i, b_i) \leftarrow_{\$} (\mathbb{Z}_q^n \times \mathbb{Z}_q)$ or $(\mathbf{a}_i, b_i) \leftarrow \mathcal{D}_{\mathbf{s}, \chi}$ for $i = 1, \dots, m$, given $n, q \geq 2$.*

The schemes considered in this thesis mainly rely on the Ring-LWE problem [Pei14], which is a variant of the LWE, where \mathbb{Z} is substituted by a ring, typically $\mathbb{Z}[X]/\langle X^n + 1 \rangle$.

NewHope

NEWHOPE [Alk+16] is a server-client key exchange protocol submitted to the NIST post-quantum standardization competition. Its protocol bases its security on a Ring Learning With Errors problem, and it informally works as follows. The server generates

a temporary pair of public and secret keys. Then it transmits the public key to the client, which uses it to encrypt a secret symmetric key. Such an encrypted key is transmitted to the server, which decrypts the ciphertext to recover the corresponding symmetric key, that can later be used for communicating. Error polynomials are used to hide the symmetric key in the ciphertext.

In the literature, there exist two variants of NEWHOPE: one is secure against chosen-plaintext attackers (CPA), and one is secure against chosen-ciphertext attackers (CCA). For our purposes, we review the basic construction of the CPA-secure NEWHOPE.

We give a more precise description of the protocol in the key generation, encryption, and decryption schemes respectively in Algorithms 2.5, 2.6, and 2.7. The relevant parameters of the scheme are the modulo q , the lattice dimension k , and the sampling parameter κ .

The routine **SampleBinomial** employs a Pseudorandom Number Generator (PRNG) that, given the initial value *seed*, outputs two random bit strings of length κ bits. The sampler then calculates the difference of the Hamming weights of both such bit strings, resulting in a binomial distributed random number. The output of **SampleBinomial** is an entire polynomial with binomially distributed coefficients. The algorithm **Encode** transforms the input message into a polynomial. It encodes each bit of the message into four coefficients. In particular, it sets such coefficients to $\{0, 0, 0, 0\}$ if the message bit is 0, otherwise, if the message bit is 1, it sets them to $\{\lfloor \frac{q}{2} \rfloor, \lfloor \frac{q}{2} \rfloor, \lfloor \frac{q}{2} \rfloor, \lfloor \frac{q}{2} \rfloor\}$.

Algorithm 2.5 NEWHOPE CPA.Keygen

Input: Public constant polynomial a

Output: Public key pk and secret key sk

- 1: $seed \xleftarrow{\$} \{0, \dots, 255\}^{32}$
 - 2: $r_1, r_2 \leftarrow \text{SampleBinomial}(seed)$
 - 3: $p \leftarrow r_1 + ar_2$
 - 4: **return** $pk = (a, p), sk = r_2$
-

Algorithm 2.6 NEWHOPE CPA.Encryption

Input: Public key (a, p) , message $\mu \in \{0, \dots, 255\}^{32}$, $coins \in \{0, \dots, 255\}^{32}$

Output: Ciphertext $A = (c_1, c_2)$

- 1: $e_1, e_2, e_3 \leftarrow \text{SampleBinomial}(coins)$
 - 2: $c_1 \leftarrow ae_1 + e_2$
 - 3: $c_2 \leftarrow pe_1 + e_3 + \text{Encode}(\mu)$
 - 4: **return** (c_1, c_2)
-

Algorithm 2.7 NEWHOPE CPA.Decryption

Input: Secret key $sk = r_2$, ciphertext $A = (c_1, c_2)$

Output: Message μ

1: **return** Decode($c_2 - c_1 r_2$)

3. Randomness optimization in Boolean masking

In this first contribution chapter, we focus on the popular Boolean masking. We discuss the challenge of improving its efficiency by providing a method to recycle a part of the randomness used in a Boolean-masked implementation.

Motivation

As described in Section 2.4, masking schemes prevent harmful power analysis attacks by protecting all sensitive information through the encoding of sensitive values. Accordingly, the designer of a masking scheme develops masked operations, the gadgets, that compute on the encodings of the inputs securely. In our study, we analyze Boolean masking, which is one of the most adopted masking schemes in practice. In this setting, masking linear operations is straightforward thanks to the linearity of the encoding function, and the main challenge is to develop secure masked non-linear gadgets, and in particular, the multiplication operation. Random values play a central role in the internal computation of the masked algorithm in order to achieve security. Belaid *et al.* showed in [Bel+16] that any t -probing secure masked multiplication requires internally at least $O(t)$ fresh random elements and, more concretely, the typical schemes for masking non-linear operations require $O(t^2)$ randoms. Additionally, the amount of randomness employed grows not only with the probing parameter t but also with the number of operations that are used by the algorithm. Since standard cryptographic algorithms typically consist of several non-linear operations, the randomness complexity of a masked cipher can be high. As an example, the AES S-box in [RP10] requires the execution of 4 multiplications and 2 refreshing schemes, for a total of hundreds masked multiplication operations in the whole block cipher.

There are two possible ways to reduce the number of random bits used in a circuit. The first method is in the spirit of the research work conducted by Belaid *et al.* in [Bel+16] and consists of designing masked non-linear operations requiring less random values. In this paper, the authors present a $t - \mathbf{NI}$ multiplication scheme at a general high order that makes use of $t + \frac{t^2}{4}$ random bits and optimized multiplication schemes for orders $t = 1, 2, 3, 4$ using respectively 2, 4 and 5 randoms. Despite the improvements achieved, the randomness complexity is reduced only by a constant factor and the most

efficient masked multiplication still requires $O(t^2)$ randomness. Moreover, this kind of strategy has two main drawbacks. First, as shown in the same paper, it has to face the natural lower bound on the amount of randomness needed to mask the non-linear operation securely. Second, such an approach does not scale well when the number of non-linear operations used in a cipher is high, as the asymptotic complexity of masking a general circuit is still $O(t^2 \cdot s)$, with s the circuit size, as in the original countermeasure in [ISW03].

A second technique to reduce the randomness cost in a circuit is to reuse random values over several masked operations. In this chapter, we explore such an alternative strategy that has gained only limited attention in the literature. Such an approach presents two main technical challenges. First, we need to ensure that when we reuse randomness among multiple operations, the composition of them does not accidentally cancel out the random bits. As an illustrative example, suppose two sensitive values a and b are masked using the same randomness r . That is, a is encoded as $(a + r, r)$ and b is encoded as $(b + r, r)$ (these may, for instance, be outputs of a masked multiplication). Now, if at some point during the computation the algorithm calculates the sum of these two encodings, then the randomness cancels out. Therefore, an adversary could access the sensitive information $a + b$ because it is not protected by any random mask. While this issue already occurs when $t = 1$, i.e., the adversary only learns one intermediate value, the situation gets much more complex when t grows, and we aim at reducing randomness between multiple masked operations. Our main contribution is to initiate the study of masking schemes where multiple gadgets share randomness and show that, despite the challenges mentioned above, amortizing randomness over multiple operations is possible and can lead in some instances to significantly more efficient masked schemes.

Contribution

Our contribution can be split in two main parts. In the first part, we analyze the generic case of reusing randomness in high-order masked implementations. In the second part, we show a particular technique that applies to the case of first-order security. As a case study, we apply our strategies to the widely known AES block cipher.

Reusing randomness for $t > 1$. We start, in Section 3.1, by considering the case of $t > 1$, i.e., when the adversary is allowed to learn multiple intermediate values. As a first contribution, we define two security notions: the weak- t -SCR, which ensures probing security when reusing randomness, and the t -SCR, which additionally provides certain composability guarantees with other gadgets, similar to the ones given by the t -NI property. We provide a composition result for our new notion and show sufficient requirements for constructing gadgets that satisfy our new notion. Our technique for sharing randomness between multiple gadgets requires to structure a circuit into so-called

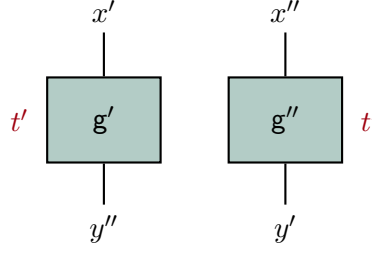


Figure 3.1.: Example of two gadgets sharing the same randomness

blocks, i.e., sets of gadgets where the individual gadgets share their random bits with a gadget in another block.

As a warm-up, we briefly present a toy example which shows in a simplified case the core idea of our contribution. As a first step, we consider two non linear gadgets g' and g'' , which internally employ the same random bits. For simplicity, in the following figures we use the same color in order to indicate that the gadgets use the same randomness. We formalize probing security in this case with the concept of *Security with Common Randomness*, as defined below for the example in Figure 3.1.

Definition 3.1 (t – SCR). *We say that the gadgets g' and g'' , whose inputs are masked by n shares, are t –Secure with Common Randomness (t – SCR in short) if*

- *each set \mathcal{P}' of t' probes on g' , and*
- *each set \mathcal{P}'' of t'' probes on g''*

such that $t' + t'' \leq t$ can be simulated by at most t' shares of the inputs of g' and t'' shares of the inputs of g'' .

If g' and g'' are t – SCR, the whole circuit in Figure 3.1 is probing secure, because, despite probing the randomness in one gadget gives information on the other gadget, the simulation is still possible without accessing the total number of input shares. In other words, the adversary does not gain any advantage by probing on both gadgets rather than only one.

As a second step, we consider the situation depicted in Figure 3.2, where g'_1 is composed with g'_2 , and g''_1 with g''_2 . The gadgets g'_1 and g''_1 share the same randomness, as well as the gadgets g'_2 and g''_2 . We call \mathcal{G}' the composition $g'_1 \circ g'_2$, and \mathcal{G}'' the composition $g''_1 \circ g''_2$. In the following lemma we show how we can compose safely gadgets that share the same randomness with other gadgets.

Lemma 3.2. *If the two blocks of gadgets \mathcal{G}' and \mathcal{G}'' depicted in Figure 3.2 and sharing the same randomness are such that*

- *g'_1 and g''_1 are t – SCR*

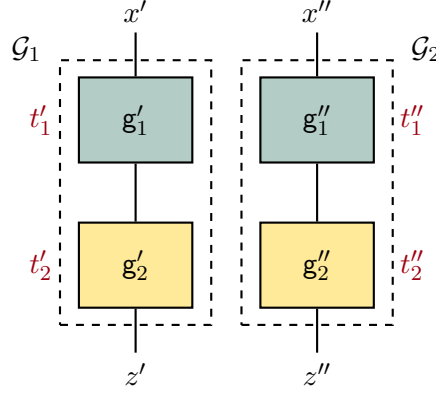


Figure 3.2.: Example of two blocks of gadgets sharing the same randomness

- g_2' and g_2'' are t -SCR

then \mathcal{G}' and \mathcal{G}'' are t -SCR.

Proof. Let t_1' be the number of probes on g_1' , t_1'' on g_1'' , t_2' on g_2' , and t_2'' on g_2'' , such that $\sum_i (t_i' + t_i'') \leq t$. We start by simulating the gadgets g_2' and g_2'' . Thanks to the t -SCR property, there exist S_2' and S_2'' two sets of indexes of the input shares of cardinality respectively t_2' and t_2'' , such that the probes on g_2' , respectively g_2'' , can be simulated by using the indexes in S_2' , resp. S_2'' .

As for the simulation of g_1' and g_1'' , since $t_1' + |S_2'| \leq t$ and $t_1'' + |S_2''| \leq t$ according to the t -SCR property, there exist S_1' and S_1'' two sets of indexes of the input shares of cardinality respectively t_1' and t_1'' such that the probes on g_1' , respectively g_1'' , can be simulated by using the indexes in S_1' , resp. S_1'' .

Combining the simulators, the whole circuit can be simulated by using the indexes in S_1' and S_2' of x' , and S_1'' and S_2'' of x'' . Therefore the simulation requires at most $t_1' + t_2'$ shares of each inputs of \mathcal{G}' and $t_1'' + t_2''$ shares of each inputs of \mathcal{G}'' . We conclude that \mathcal{G}' and \mathcal{G}'' are t -SCR. \square

We point out that the blocks of gadgets \mathcal{G}' and \mathcal{G}'' could be composed with each other in a bigger circuit, by additionally ensuring that they are t -SNI.

In the rest of the chapter, we will show how the technique just presented can scale to a more general situation, where a circuit is divided into blocks of gadgets with shared randomness. We will see that the randomness of a circuit cannot be totally reused. Indeed, a crucial requirement to securely reuse randomness is independence among the inputs of these aforementioned blocks of gadgets. To this end, the last gadget in each of these blocks needs to be refreshed. For the same reason, it is necessary to put

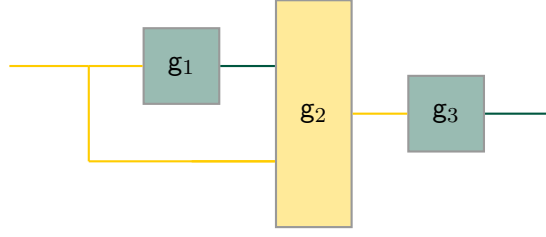


Figure 3.3.: Example of alternate reuse of randomness at 1st order

an extra refreshing scheme whenever the inputs of gadgets sharing randomness have dependent masks.

Reusing randomness for $t = 1$. We design new schemes for multiplication and addition that achieve probing security at first order and employ only 2 random bits. The result holds for any arbitrary complex masked circuits. We remind that the multiplication algorithm secure at first order requires only one random bit. Our approach consists in using the two random bits alternatively in the schemes of a circuit, in such a way that two consecutive gadgets use a different random bit. Notice that, since randomness can cancel out when used several times in the same circuit, such a scheme needs to be designed carefully. Moreover, the security analysis cannot easily rely on the compositional notion of 1-SNI [Bar+15a]. Indeed, such notions require that each gadget employ independent randomness. A graphical example of the idea is Figure 3.3, where different colors represent dependence on different random bits.

Implementation results. We finally complete our analysis with a case study by applying our new countermeasures to masking the AES algorithm. Our analysis shows that for orders up to $t = 3$ we can not only significantly reduce the amount of randomness needed, but also improve on efficiency.

Related work

Despite being a major practical bottleneck, reducing the amount of randomness in masking schemes has not been the focus of many research works until 2017. Belaid *et al.* in [Bel+16] presented constructions that reduce the number of random bits needed in a masked multiplication. Additionally, the authors gave a lower bound on the minimal amount of randomness required to protect a masked multiplication. However, the best-known scheme still requires an amount of randomness quadratic in the security parameter. Moreover, a strategy for reducing the randomness complexity of first-order threshold implementations of Keccak was examined in [Bil+13].

From a practical point of view, the concept of “recycled” randomness was briefly investigated in [Bal+14]. The authors practically evaluated the influence of reusing part of the masks in some case studies and concluded that, in some circumstances, this causes a security decrease. Nevertheless, these results lacked formal security proofs and therefore do not negatively reflect on our methodology.

From a theoretical point of view, Ishai *et al.* [Ish+13] showed that any circuit can be masked using an amount of randomness which is independent of the size of the circuit. More precisely, a circuit can be protected using polynomial in t random bits. The constructions proposed are based on bipartite expander graphs and are mainly interesting from a theoretical point of view, since they become meaningful only when t is very large. On the contrary, in our work, we focused on the cases of small t , which are more relevant in practice.

After the publication of our work, other papers with the scope of amortizing randomness in masked implementations have been published. De Meyer *et al.*, in their paper on implementing the AES using multiplicative masks [DRB18], presented a method for recycling randomness in the circuit computing the Kronecker delta function². The strategy improves the randomness requirement from 7 to 3 bits at first-order and from 21 to 13 bits for second-order of security. However, the method is tailored specifically for this circuit and not easy to generalize.

In the same year, Wegener and Moradi [WM18] showed a first-order secure implementation of AES, which does not make use of fresh randomness. The methodology is valid for any bijective S-box. Nevertheless, it considers masking in four-shares and does not provide any security proofs, but only a practical analysis.

Lately, following the work of Ishai *et al.* [Ish+13], Coron *et al.* presented in [CGZ19] a construction which reduces the number of calls to the TRNG employing a pseudorandom generator that generates all the randomness used by the circuit. According to their construction, masking the AES requires only 48 bytes of randomness to get second-order security and 108 to get third-order security, instead of respectively 1415 and 2530, required by our scheme.

3.1. Reusing randomness in private circuits

We start by analyzing privacy of a particular set of gadgets $\mathbf{g}_1, \dots, \mathbf{g}_d$ in which the random component is substituted by a set of bits $\mathbf{r} = (r_1, \dots, r_l)$ taken at random, but reused by each of the gadgets $\mathbf{g}_1, \dots, \mathbf{g}_d$.

In the following we indicate with $\mathbf{g}(\mathbf{x}, \mathbf{r})$ a gadget which takes as input a mask \mathbf{x} and internally uses a vector \mathbf{r} of random bits, where \mathbf{r} is of the form $(\mathbf{r}_1, \dots, \mathbf{r}_n)$ and each \mathbf{r}_i is the vector of the random bits involved in the computation of the i^{th} output share. For

²The Kronecker delta function outputs 1 when its input is 0, and outputs 0 otherwise.

example, referring to Algorithm 3.4, \mathbf{r}_1 is the vector (r_1, r_7, r_{13}, r_8) . If the input values are already clear from the context, in the rest of this chapter we will mainly specify a gadget with only its random component \mathbf{r} , so it will be indicated as $\mathbf{g}(\mathbf{r})$. Moreover, we suppose that all the gadgets $\mathbf{g}(\mathbf{r})$ are such that every intermediate value used in the computation of the i^{th} output share contains only random bits in \mathbf{r}_i .

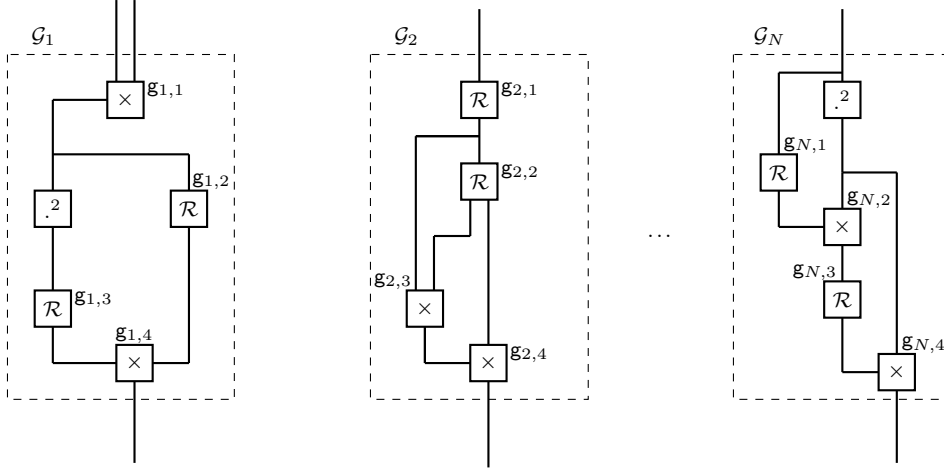
In order to analyze the possibility of randomness reuse, we first introduce the definition of t -Security-with-Common-Randomness, which formalizes the requirements to guarantee security in a situation where the random values are shared among the gadgets. First, we give in Definition 3.3 a weaker form of the new notion, which ensures probing security, but it cannot be used when gadgets compose in a more complicated circuit. Then, we give in Definition 3.4 a stronger notion that allows us to make security arguments on the composition of t -SCR gadgets.

Definition 3.3 (Weak t -SCR). *Let \mathbf{r} be a set of random bits. We say that the gadgets $\mathbf{g}_1(\mathbf{r}), \dots, \mathbf{g}_d(\mathbf{r})$ receiving each m inputs split into n shares are t -Secure with Common Randomness (t -SCR in short) if for each set \mathcal{P}_i of t_i probes on \mathbf{g}_i such that $\sum_{i=1}^d t_i \leq t$, the probes in \mathcal{P}_i can be simulated by at most $n-1$ shares of the input of \mathbf{g}_i and the simulation is consistent with the shared random component.*

Definition 3.4 (t -SCR). *Let \mathbf{r} be a set of random bits. We say that the gadgets $\mathbf{g}_1(\mathbf{r}), \dots, \mathbf{g}_d(\mathbf{r})$ receiving each m inputs split into n shares are t -Secure with Common Randomness (t -SCR in short) if for each set \mathcal{P}_i of t_i probes on \mathbf{g}_i such that $\sum_{i=1}^d t_i \leq t$, the probes in \mathcal{P}_i can be simulated by at most t_i shares of the input of \mathbf{g}_i and the simulation is consistent with the shared random component.*

We point out that the definition of *weak t -SCR* exactly corresponds to the definition of probing security, while the definition of t -SCR is the equivalent of the definition of t -NI, considering a circuit formed by the $\mathbf{g}_1(\mathbf{r}), \dots, \mathbf{g}_d(\mathbf{r})$ gadgets.

We now use this definition in order to analyze the randomness reuse among entire regions of the circuit, which we refer to as *blocks*. More formally, with the term *block of gadgets*, we define a sub-circuit composed by gadgets, with input an encoding of a certain x and output an encoding of y . Since our analysis focuses on the randomness, when we refer to such a block, we only consider the randomized gadgets. In particular, we indicate a block of gadgets as $\mathcal{G}(\mathbf{R}) = \{\mathbf{g}_1(\mathbf{r}_1), \dots, \mathbf{g}_d(\mathbf{r}_d)\}$, where the \mathbf{g}_i represent the randomized gadgets in the block and $\mathbf{R} = (\mathbf{r}_1, \dots, \mathbf{r}_d)$ constitutes the total amount of randomness employed by \mathcal{G} . We assume without loss of generality that the input of such a \mathcal{G} is the input of the first randomized gadget \mathbf{g}_1 . Indeed, even if the first gadget of the block was a non-randomized one (i.e., a linear gadget), then this would change the value of the input, but not its properties related to the independence. We call *dimension* of a block \mathcal{G} the number of randomized gadgets \mathbf{g}_i composing the block. In Figure 3.4, it is depicted an example of N blocks of gadgets of dimension 4 each.


 Figure 3.4.: An example of N blocks of gadgets of dimension $d = 4$ each

The first result that we show about the block of gadgets aforementioned is the following lemma, which gives a simple compositional result for multiple blocks of gadgets, where each of such blocks uses the same random component \mathbf{R} . Informally speaking, let \mathcal{G}_j be multiple sets of gadgets, where all gadgets in \mathcal{G}_j share the same randomness. Then, the lemma below shows that if the gadgets in \mathcal{G}_j are t -SCR, then also the composition of the gadgets in all sets \mathcal{G}_j are t -SCR. We underline that such a block constitutes itself a gadget. For simplicity, we assume that the blocks of gadgets that we consider in the lemma below all have the same dimension d . However, our analysis can be easily generalized to a setting where each block has a different dimension.

Lemma 3.5. *Consider N blocks of gadgets*

$$\begin{aligned} \mathcal{G}_1(\mathbf{R}) &= \{g_{1,1}(\mathbf{r}_1), \dots, g_{1,d}(\mathbf{r}_d)\} \\ &\vdots \\ \mathcal{G}_N(\mathbf{R}) &= \{g_{N,1}(\mathbf{r}_1), \dots, g_{N,d}(\mathbf{r}_d)\} \end{aligned}$$

sharing the same random component $\mathbf{R} = (\mathbf{r}_1, \dots, \mathbf{r}_d)$.

If for all $j = 1, \dots, d$ the gadgets $\{g_{1,j}(\mathbf{r}_j), \dots, g_{N,j}(\mathbf{r}_j)\}$ are t -SCR and each gadget is t -SNI, then the blocks of gadgets $\{\mathcal{G}_1, \dots, \mathcal{G}_N\}$ are t -SCR as well.

Proof. For each $i = 1, \dots, N$ and each $j = 1, \dots, d$ let t_j^i be the number of probes on the gadget $g_{i,j}$. We can prove the statement with an inductive argument on the dimension of the blocks $j = 1, \dots, d$.

- If $j = d$, then by hypothesis $\{g_{1,d}, \dots, g_{N,d}\}$ are t -SCR, i.e., for each $i = 1, \dots, N$ there exists a set of indexes S_d^i , with $|S_d^i| \leq t_d^i$, such that for each $g_{i,d}$ the adversary's view can be simulated by using the input shares with indexes in S_d^i .

- If $j < d$ and $\{\{\mathbf{g}_{1,j+1}, \dots, \mathbf{g}_{N,j+1}\}, \dots, \{\mathbf{g}_{1,d}, \dots, \mathbf{g}_{N,d}\}\}$ are t -SCR, then there exists a set of indexes S_{j+1}^i , with $|S_{j+1}^i| \leq t_{j+1}^i$, such that for each $\mathbf{g}_{i,j+1} \circ \dots \circ \mathbf{g}_{i,d}$ the adversary's view can be simulated by using the indexes in S_{j+1}^i . Since for each i we have that $t_j^i + |S_{j+1}^i| \leq t_j^i + t_{j+1}^i \leq t$ and $\mathbf{g}_{i,j}$ is t -SNI, the simulation of $\{\mathbf{g}_{1,j}, \dots, \mathbf{g}_{N,j}\}$ is independent from each $|S_{j+1}^i|$. Since by hypothesis $\{\mathbf{g}_{1,j}, \dots, \mathbf{g}_{N,j}\}$ are t -SCR, for each $i = 1, \dots, N$ there exists a set of indexes S_j^i , with $|S_j^i| \leq t_j^i$, such that for each $\mathbf{g}_{i,d}$ the adversary's view can be simulated by using the input shares with indexes in S_j^i .

We conclude that for every dimension d of the blocks, the set $\{\mathcal{G}_1, \dots, \mathcal{G}_N\}$ is t -SCR. \square

As a next step we want now to use the result above in order to build circuits composed by block of gadgets sharing the same randomness. We first point out that the t -SCR property itself is not sufficient to guarantee a sound composition. Therefore, when used in combination with other gadgets, a t -SCR scheme needs additionally to satisfy the t -SNI property. We summarize this observation in the following theorem, which gives a global result for circuits designed in blocks of gadgets sharing the same randomness.

Theorem 3.6. *Let \mathcal{C} be a circuit composed by N blocks of gadgets $\mathcal{G}_1(\mathbf{R}), \dots, \mathcal{G}_N(\mathbf{R})$ where $\mathcal{G}_i(\mathbf{R}) = \{\mathbf{g}_{i,1}(\mathbf{r}_1), \dots, \mathbf{g}_{i,d}(\mathbf{r}_d)\}$ for each $i = 1, \dots, N$ and with inputs encoded in n shares and such that the gadgets outside such blocks are either linear or t -SNI. If*

- $\forall i = 1, \dots, N$ \mathcal{G}_i is t -SNI and
- $\forall j = 1, \dots, d$ $\mathbf{g}_{1,j}, \dots, \mathbf{g}_{N,j}$ are t -SCR

then the circuit \mathcal{C} is t -probing secure.

Proof. The proof of the theorem is straightforward. Indeed, Lemma 3.5 implies that $\mathcal{G}_1, \dots, \mathcal{G}_N$ are t -SCR. Moreover, we point out that since the \mathcal{G}_i is t -SNI, for every $i = 1, \dots, N$, the composition is secure

- among the blocks \mathcal{G}_i
- of the \mathcal{G}_i with other randomized and t -SNI gadgets using fresh randomness

This is sufficient to prove that the circuit \mathcal{C} is t probing secure. \square

To sum up, we showed in this section that, under certain conditions, it is possible to design a circuit which internally reuses the random bits involved and remains probing secure. Therefore, if used appropriately, this result allows us to decrease the amount of randomness necessary in order to have a private circuit (because all the blocks share the same randomness). Nevertheless, we remark that in practice, in order to have blocks of t -SCR gadgets, the inputs of the first gadget of each block must be mutually independent. Indeed, otherwise, it would not be always possible to have the independent simulation required by Definition 3.4. Our analysis needs another step to determine how to fulfill this requirement in the presence of not naturally independent inputs.

Algorithm 3.1 Refreshing gadget **Ind**

Input: a_1, \dots, a_n such that $\bigoplus a_i = a$

Output: $a_1^{in}, \dots, a_n^{in}$ such that $\bigoplus a_i^{in} = a$

- 1: $(a_1^{in}, \dots, a_n^{in}) \leftarrow (a_1, \dots, a_n);$
 - 2: **for** $i = 2$ **to** n **do**
 - 3: $r_i \xleftarrow{\$} \mathbb{F}_2^n;$
 - 4: $a_i^{in} \leftarrow a_i^{in} + r_i;$
 - 5: $a_1^{in} \leftarrow a_1^{in} + r_i;$
-

The Ind gadget

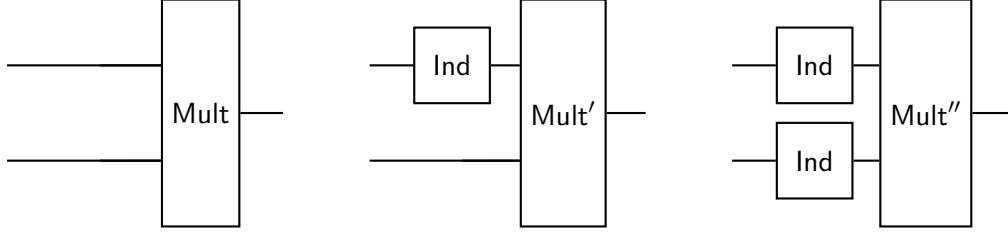
In this section, we present a gadget, that we call **Ind**, which, given an input x , produces an output $\text{Ind}(x)$ independent from x . Thanks to this property, **Ind** can be adopted in order to create independence between inputs of gadgets sharing the same randomness. The algorithm, already proposed in [Cor+13] as a refreshing scheme, is depicted in Algorithm 3.1, and it requires the use of n fresh random bits, where n is the number of input shares. It was proven to be $t - \text{NI}$ and not $t - \text{SNI}$ in [Bar+15a], and therefore it must be used carefully, because it is not always securely composable with other gadgets.

As an example, the scheme **Ind** can be applied at the inputs of the multiplication schemes using the same randomness in order to make them independent of each other. Whenever the scheme cannot be used because leading to an insecure circuit, the gadgets with dependent inputs cannot share the same randomness.

Therefore, when designing such circuits, even if on the one hand the randomness involved in the gadgets can be reused entirely, on the other hand additional **Ind** schemes are often required to guarantee the independence of the inputs of gadgets sharing randomness.

Let us consider, for example, the case when we aim at reusing the randomness among N multiplication schemes. Let $t = 2$, $n = 3$, and let the number of random bits per scheme be 3 (this is a reasonable supposition since the most efficient known $t - \text{SNI}$ multiplication scheme uses 3 random bits). If the inputs of the multiplication schemes are dependent to each other, in order to have independent inputs and securely reuse the randomness, we need to add two **Ind** gadgets, one per each input, as for **Mult''** in Figure 3.5. It is easy to check that the number of fresh random bits injected in the circuit by the **Ind** gadgets exceeds the one that it would be used if the multiplication schemes would not reuse the same randomness. On the contrary, if only one of the two inputs of the multiplication schemes sharing randomness is dependent, we need only one **Ind** scheme per multiplication, as for **Mult'** in Figure 3.5. In this case, reusing randomness is convenient.

The examples above show that the process of reusing randomness is not straightforward. Only a good trade-off between the fresh randomness used by the **Ind** gadgets and the amount of randomness saved by sharing it gives actual efficiency. Therefore, it is important


 Figure 3.5.: Example of gadgets with different use of the `Ind` refreshing scheme

to make a careful analysis of the circuit structure before applying our method. First, to understand if the randomness cost is actually amortized, and secondly, to check that the security still holds, even when employing the `Ind` gadgets. As we will see later in Section 3.5, for circuits with an obvious structure (e.g., layers for symmetric ciphers), which contain easily-exploitable regularities, this is not a hard task, and the optimal solutions can be usually found quickly.

3.2. A t – SCR multiplication Scheme

Now that we have set the basic theory about the reuse of randomness in a circuit, we can show how to construct the basic randomized gadgets, i.e., multiplication and refreshing, such that they satisfy the t – SCR property.

We start from the multiplication schemes. We notice that it is possible to build multiplication schemes providing t – SCR security, by satisfying two essential properties, i.e., $\lfloor \frac{t}{2} \rfloor$ -non-completeness and t – SNI. Then, we discuss how to construct instantiations of our multiplication according to these properties, and we finally provide their security proof.

First, we construct a multiplication scheme, $\text{Mult} : \mathbb{F}_2^n \times \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$, which is $\lfloor \frac{t}{2} \rfloor$ -non-complete. We point out that when $n = t + 1$, this property ensures the t – SCR notion, as defined in Definition 3.4. On the other hand, if $n > t + 1$ the property guarantees only the weak t – SCR from Definition 3.3. We provide the formal security analysis in Section 3.2.

The process for constructing a multiplication scheme which is $\lfloor \frac{t}{2} \rfloor$ -non-complete is similar to the one of finding a $\lfloor \frac{t}{2} \rfloor$ -order TI of the AND-gate [Rep+15] or multiplication [Cnu+15]. For our application, we additionally need that the number of output shares is equal to the number of input shares. Most higher-order TI avoid this restriction with additional refreshing- and compression-layers. In our case, we aim at fulfilling $\lfloor \frac{t}{2} \rfloor$ -non-completeness without fresh randomness, and therefore we cannot rely on the compression of the output shares. Unfortunately, this is only possible for very specific values of n . Due to this minor difference, we cannot directly use the bounds from the original publications related to higher-order TI. In the following, we derive an

equation for n given an arbitrary t for which there exist a $\lfloor \frac{t}{2} \rfloor$ -non-complete Mult.

Let l be the number of input shares that are leaked by each of the output shares. We notice that in order to fulfill the $\lfloor \frac{t}{2} \rfloor$ -non-completeness, even combining $\lfloor \frac{t}{2} \rfloor$ output shares must be independent of one input share. Therefore, we can construct a scheme with the above property only for values of n given by

$$\left\lfloor \frac{t}{2} \right\rfloor \cdot l + 1 = n \quad (3.1)$$

To construct a $\lfloor \frac{t}{2} \rfloor$ -non-complete multiplication algorithm, we need to distribute $\binom{n}{2}$ terms of the form $a_i b_j + a_j b_i$ over n output shares, i.e., each output share is made up of the sum of $\frac{n-1}{2}$ terms. Each of these terms leaks information about the tuples (a_i, a_j) and (b_i, b_j) , assuming that the encodings a and b are independent and randomly chosen. For a given l , the maximum number of possible terms, which can be combined without leaking about more than l shares of a or b , is $\binom{l}{2}$. The remaining $a_i b_i$ are equally distributed over the output shares without increasing l . By combining these two observations, we conclude that

$$\frac{n-1}{2} = \frac{l^2-l}{2}. \quad (3.2)$$

Based on Equation (3.1), the minimum number of shares for $\lfloor \frac{t}{2} \rfloor$ -non-completeness is $n = \lfloor \frac{t}{2} \rfloor \cdot l + 1$. We combine this with Equation (3.2) and derive

$$n = \left\lfloor \frac{t}{2} \right\rfloor^2 + \left\lfloor \frac{t}{2} \right\rfloor + 1. \quad (3.3)$$

We use Equation (3.3) to compute the number of shares for our t -SCR multiplication scheme with $t \geq 4$. For $t \leq 3$, the number of shares is bounded by the requirement for the multiplication to be t -SNI, i.e., $n > t$.

From the equation above, we can see that it is not possible to find a $\lfloor \frac{t}{2} \rfloor$ -non-complete multiplication scheme with $n = t + 1$ for values of $t \geq 4$. This means that for $t \geq 4$ we can only provide a weakly t -SCR multiplication scheme.

To achieve t -SCR, it is necessary to employ randomness in the calculations. Initially, $\frac{n^2}{2}$ random components r_i need to be added for the multiplication to be t -SNI. We add a subset of n random components to each output share, and we equally distribute it over the sum. In order to ensure the simulatability of the gadget by using a limited number of input shares, as required by the definition of t -SNI, each random bit is involved a second time in the computation of a single different output share. In this way, the distribution of the random bits allows simulating the output probes with a random and independent value.

Particular attention has to be reserved for odd orders. Indeed, when dealing with odd orders, the two properties aforementioned are not sufficient for guaranteeing privacy in

Algorithm 3.2 Mult² for order $t = 2$ with $n = 3$ shares

Input: shares a_1, \dots, a_3 such that $\bigoplus a_i = a$, shares b_1, \dots, b_3 such that $\bigoplus b_i = b$

Output: shares c_1, \dots, c_3 such that $\bigoplus c_i = a \cdot b$

- 1: $c_1 = (a_1 b_2 + r_1) + (a_2 b_1 + r_2) + a_2 b_2$;
 - 2: $c_2 = (a_2 b_3 + r_2) + (a_3 b_2 + r_3) + a_3 b_3$;
 - 3: $c_3 = (a_3 b_1 + r_3) + (a_1 b_3 + r_1) + a_1 b_1$;
-

our scenarios. Informally speaking, the $\lfloor \frac{t}{2} \rfloor$ -non-completeness allows probing up to $\lfloor \frac{t}{2} \rfloor$ of the values in two different gadgets sharing the same randomness, without that the cancellation of the random bits let the adversary know more than $n - 1$ shares of the secret. However, when t is odd, the adversary can also have another probe with the same randomness, in addition to the $\lfloor \frac{t}{2} \rfloor$ probes per gadget sharing the same randomness. This probe might add at least another input share to the knowledge of the adversary. Consequently, the scheme needs to satisfy a third property, the *Special Non-Completeness*.

Definition 3.7 (Special Non-Completeness). *For every set of at most t probes p_1, \dots, p_k , q_1, \dots, q_h such that*

- p_1, \dots, p_k are probes on the output shares depending respectively on the vectors of random bit $\mathbf{r}_{p_1}, \dots, \mathbf{r}_{p_k}$, where $k \leq \frac{t}{2}$,
- q_1, \dots, q_h are probes on the internal values depending respectively on the vectors of random bit $\mathbf{r}_{q_1}, \dots, \mathbf{r}_{q_h}$, where $h \leq t - 2 \cdot k$,
- $\forall i \in [1, h], \exists j \in [1, k]$ such that $\mathbf{r}_{q_i} \subseteq \mathbf{r}_{p_j}$,

the set $\{p_1, \dots, p_k, q_1, \dots, q_h\}$ is non-complete without randomness.

The construction of a t – SCR multiplication scheme following the aforementioned guidelines is not always easy. For increasing values of t , finding a distribution of terms that fulfills $\lfloor \frac{t}{2} \rfloor$ -non-completeness or special non-completeness becomes a complex task, due to a large number of possible combinations. For $t \leq 5$, possible t – SCR multiplication schemes are defined respectively in Algorithms 3.2, 3.3, 3.4 and 3.5. We stress that Algorithms 3.2, 3.3 achieve the stronger version of t – SCR, while Algorithms 3.4 and 3.5 are only weak t – SCR. We recall that the computation in all the algorithms follows from left to right, and the operations in brackets are executed first.

Security analysis

In this section we present the security analysis of the multiplication schemes, constructed according to the properties given in the previous section. We highlight that the schemes in Algorithm 3.2 and 3.3 were both already proven to be t – SNI in [Bar+15a]. We prove the schemes for orders 2 and 3 to be t – SCR according to Definition 3.4 and we

3. Randomness optimization in Boolean masking

Algorithm 3.3 Mult^3 for order $t = 3$ with $n = 4$ shares

Input: shares a_1, \dots, a_4 such that $\bigoplus a_i = a$, shares b_1, \dots, b_4 such that $\bigoplus b_i = b$

Output: shares c_1, \dots, c_4 such that $\bigoplus c_i = a \cdot b$

- 1: $c_1 = (a_1b_2 + r_1) + (a_3b_1 + r_5) + (a_3b_2 + r_4) + a_1b_1$;
 - 2: $c_2 = (a_2b_1 + r_1) + (a_4b_2 + r_6) + (a_4b_1 + r_2) + a_2b_2$;
 - 3: $c_3 = (a_3b_4 + r_3) + (a_2b_3 + r_5) + (a_2b_4 + r_2) + a_3b_3$;
 - 4: $c_4 = (a_4b_3 + r_3) + (a_1b_4 + r_6) + (a_1b_3 + r_4) + a_4b_4$;
-

Algorithm 3.4 Mult^4 for order $t = 4$ with $n = 7$ shares

Input: shares a_1, \dots, a_7 such that $\bigoplus a_i = a$, shares b_1, \dots, b_7 such that $\bigoplus b_i = b$

Output: shares c_1, \dots, c_7 such that $\bigoplus c_i = a \cdot b$

- 1: $c_1 = r_{15} + a_1b_1 + r_1 + a_1b_2 + a_2b_1 + r_{18} + a_1b_3 + a_3b_1 + r_7 + a_2b_3 + a_3b_2 + r_{13} + r_8$;
 - 2: $c_2 = r_{16} + a_2b_2 + r_2 + a_2b_4 + a_4b_2 + r_{17} + a_2b_6 + a_6b_2 + r_1 + a_4b_6 + a_6b_4 + r_{14}$;
 - 3: $c_3 = r_{17} + a_4b_4 + r_8 + a_4b_1 + a_1b_4 + r_{10} + a_4b_5 + a_5b_4 + r_2 + a_1b_5 + a_5b_1 + r_3$;
 - 4: $c_4 = r_{18} + a_5b_5 + r_9 + a_5b_2 + a_2b_5 + r_{16} + a_5b_7 + a_7b_5 + r_4 + a_2b_7 + a_7b_2 + r_{11}$;
 - 5: $c_5 = r_5 + a_7b_7 + r_4 + a_7b_1 + a_1b_7 + r_{15} + a_7b_6 + a_6b_7 + r_{12} + a_1b_6 + a_6b_1 + r_3$;
 - 6: $c_6 = r_6 + a_6b_6 + r_{13} + a_6b_3 + a_3b_6 + r_5 + a_6b_5 + a_5b_6 + r_9 + a_3b_5 + a_5b_3 + r_{11}$;
 - 7: $c_7 = r_{10} + a_3b_3 + r_{12} + a_3b_4 + a_4b_3 + r_6 + a_3b_7 + a_7b_3 + r_{14} + a_4b_7 + a_7b_4 + r_7$;
-

prove the scheme for orders 4 and 5 to be both $t - \text{SNI}$ and weak $t - \text{SCR}$, according to Definition 3.3.

The proofs are all structured similarly. As a first step, we list all possible probes. Secondly, for each set of at most t possible probes, we construct sets of indexes of input shares. The third and last step consists of providing a simulation of the probes using only the input shares with the index contained in such sets. The crucial point of the proof is showing that the simulation uses a number of input shares corresponding to the requirements of the property that we are proving. So, for instance, in the case of proofs of $t - \text{SNI}$, the simulation should require a number of input shares that is less or equal to the number of internal probes. We remind that, as explained in Section 2.6 and Lemma 2.3, showing the simulatability according to the definitions of t -probing security and $t - \text{NI}$ is equivalent to show that the distributions of the simulation and the t -limited attack are identical.

In the rest of the section, N is a natural number, with $N \geq 2$, and it represents the number of gadgets sharing the same randomness.

Proposition 3.8 ($t - \text{SCR}$ for order $t = 2$). *Let $\text{Mult}_1^2, \dots, \text{Mult}_N^2$ be a set of N multiplication schemes as in Algorithm 3.2, with inputs $(\mathbf{a}^{(1)}, \mathbf{b}^{(1)}), \dots, (\mathbf{a}^{(N)}, \mathbf{b}^{(N)})$ and outputs the values $\mathbf{c}^{(1)}, \dots, \mathbf{c}^{(N)}$. Suppose that the maskings of the inputs are independent and uniformly chosen and that for $k = 1, \dots, N$ each Mult_k^2 uses the same random bits r_1, r_2, r_3 .*

Algorithm 3.5 Mult^5 for order $t = 5$ with $n = 7$ shares

Input: shares a_1, \dots, a_7 such that $\bigoplus a_i = a$, shares b_1, \dots, b_7 such that $\bigoplus b_i = b$

Output: shares c_1, \dots, c_7 such that $\bigoplus c_i = a \cdot b$

- 1: $c_1 = (a_2b_3 + r_1) + (a_3b_2 + r_7) + (a_1b_3 + r_8) + (a_3b_1 + r_{13}) + (a_2b_1 + r_{15}) + (a_1b_2 + r_{19}) + a_1b_1;$
 - 2: $c_2 = (a_4b_2 + r_2) + (a_2b_4 + r_1) + (a_6b_2 + r_9) + (a_4b_6 + r_{14}) + (a_2b_6 + r_{16}) + (a_6b_4 + r_{20}) + a_2b_2;$
 - 3: $c_3 = (a_5b_4 + r_3) + (a_1b_4 + r_2) + (a_4b_5 + r_{10}) + (a_1b_5 + r_8) + (a_5b_1 + r_{17}) + (a_4b_1 + r_{21}) + a_4b_4;$
 - 4: $c_4 = (a_2b_7 + r_4) + (a_5b_7 + r_3) + (a_5b_2 + r_{11}) + (a_2b_5 + r_9) + (a_7b_2 + r_{18}) + (a_7b_5 + r_{15}) + a_5b_5;$
 - 5: $c_5 = (a_6b_7 + r_5) + (a_7b_6 + r_4) + (a_7b_1 + r_{12}) + (a_1b_7 + r_{10}) + (a_1b_6 + r_{19}) + (a_6b_1 + r_{16}) + a_7b_7;$
 - 6: $c_6 = (a_3b_6 + r_6) + (a_6b_3 + r_5) + (a_5b_3 + r_{13}) + (a_5b_6 + r_{11}) + (a_6b_5 + r_{20}) + (a_3b_5 + r_{17}) + a_6b_6;$
 - 7: $c_7 = (a_3b_4 + r_7) + (a_7b_4 + r_{14}) + (a_3b_7 + r_6) + (a_4b_7 + r_{12}) + (a_7b_3 + r_{21}) + (a_4b_3 + r_{18}) + a_3b_3;$
-

Then $\text{Mult}_1^2, \dots, \text{Mult}_N^2$ are 2 – SCR.

Proof. Let $\Omega = (\mathcal{P}_1, \dots, \mathcal{P}_N)$ be a set of 2 observations respectively on $\text{Mult}_1^2, \dots, \text{Mult}_N^2$. We show that $\text{Mult}_1^2, \dots, \text{Mult}_N^2$ satisfy Definition 3.4, i.e., that the probes in \mathcal{P}_i can be consistently simulated by at most $|\mathcal{P}_i|$ shares of the inputs of Mult_i^2 .

Probes classification.

We classify all the possible probes in the following groups, for $i, j = 1, 2, 3$:

- (1) r_1, r_2, r_3
- (2) $a_i^{(k)}, b_j^{(k)}, a_i^{(k)} \cdot b_j^{(k)}$
- (3) $(a_i^{(k)} \cdot b_j^{(k)} + r_h)$
- (4) $(a_i^{(k)} \cdot b_j^{(k)} + r_h) + (a_i^{(k)} \cdot b_j^{(k)} + r_l)$
- (5) $c_i^{(k)}$

Construction of sets of shares indexes.

We construct the sets of input shares $I^{(k)}, J^{(k)}$ by adding i to $I^{(k)}$ and j to $J^{(k)}$, for each $k = 1, \dots, N$. Note that $|I^{(k)}|, |J^{(k)}| \leq 2$.

Simulation.

The simulation starts by defining \mathcal{R} the set of all random bits appearing in the adversarial's observations and simulating each of them uniformly at random. Then it proceeds as follows:

- For each probe in group (2), simulate the value using the shares of the inputs with index in $I^{(k)}$ and $J^{(k)}$.
- For each probe in group (3), if $r_h \in \mathcal{R}$ or $\exists k$ such that $a_i^{(k)} \cdot b_j^{(k)} + r_h \in \Omega$, use the shares of the inputs with index $i \in I^{(k)}$ and $j \in J^{(k)}$. Otherwise, simulate the probe with a uniformly random value.
- For each probe in group (4), if $\exists \hat{k}$ such that $(a_i^{(\hat{k})} \cdot b_j^{(\hat{k})} + r_h) + (a_j^{(\hat{k})} \cdot b_i^{(\hat{k})} + r_l) \in \Omega$, then use the shares of the inputs with index in $I^{(k)}$ and $J^{(k)}$ and the same random bits r_h, r_l ; otherwise, since for any other kind of probe there will always be a random bit not observed, we can simulate the probe with a uniformly random value.

- For each probe in group (5), proceed similarly as in the previous step. If $\exists \hat{k}$ such that $c_i^{(\hat{k})} \in \Omega$, then use the shares of the inputs with index in $I^{(\hat{k})}$ and $J^{(\hat{k})}$ and the same random bits r_h, r_l and note that each product $a_i^{(\hat{k})} \cdot b_i^{(\hat{k})}$ does not add any index to $I^{(\hat{k})}$ and $J^{(\hat{k})}$. Otherwise we can simulate the probe with a uniformly random value.

Since for each case the simulation uses only the shares of the inputs with indexes in $I^{(k)}, J^{(k)}$, and since $|I^{(k)}|, |J^{(k)}| \leq 2$, the simulation requires at most 2 shares per each input. Therefore $\text{Mult}_1^2, \dots, \text{Mult}_N^2$ is 2-SCR. \square

Proposition 3.9 (t -SCR for order $t = 3$). *Let $\text{Mult}_1^3, \dots, \text{Mult}_N^3$ be a set of N multiplication schemes as in Algorithm 3.3, with inputs $(\mathbf{a}^{(1)}, \mathbf{b}^{(1)}), \dots, (\mathbf{a}^{(N)}, \mathbf{b}^{(N)})$ and outputs the values $\mathbf{c}^{(1)}, \dots, \mathbf{c}^{(N)}$. Suppose that the maskings of the inputs are independent and uniformly chosen and that for $k = 1, \dots, N$ each Mult_k^3 uses the same random bits r_1, r_2, r_3, r_4 . Then $\text{Mult}_1^3, \dots, \text{Mult}_N^3$ are 3-SCR.*

Proof. Let $\Omega = (\mathcal{P}_1, \dots, \mathcal{P}_N)$ be a set of 3 observations respectively on $\text{Mult}_1^3, \dots, \text{Mult}_N^3$. We show that $\text{Mult}_1^3, \dots, \text{Mult}_N^3$ satisfy Definition 3.4, i.e., that the probes in \mathcal{P}_i can be consistently simulated by at most $|\mathcal{P}_i|$ shares of the inputs of Mult_i^3 .

Probes classification.

We classify all the possible probes in the following groups, for $i, j = 1, \dots, 4$:

- (1) $r_1, r_2, r_3, r_4, r_5, r_6$
- (2) $a_i^{(k)}, b_j^{(k)}, a_i^{(k)} \cdot b_j^{(k)}$
- (3) $(a_i^{(k)} \cdot b_j^{(k)} + r_h)$
- (4) $(a_i^{(k)} \cdot b_j^{(k)} + r_h) + (a_i^{(k)} \cdot b_j^{(k)} + r_l)$
- (5) $(a_i^{(k)} \cdot b_j^{(k)} + r_h) + (a_i^{(k)} \cdot b_j^{(k)} + r_l) + (a_i^{(k)} \cdot b_j^{(k)} + r_m)$
- (6) $c_i^{(k)}$

Construction of sets of shares indexes.

We construct the sets of input shares $I^{(k)}, J^{(k)}$ by adding i to $I^{(k)}$ and j to $J^{(k)}$, for each $k = 1, \dots, N$. Note that $|I^{(k)}|, |J^{(k)}| \leq 3$.

Simulation.

The simulation starts by defining \mathcal{R} the set of all the random bits appearing in the adversarial's observations and simulating each of them uniformly at random. Then it proceeds as follows:

- For each probe in group (2) and (3), follow the same procedure of the proof for Proposition 3.8.
- For each probe in group (4), if $\exists \hat{k}$ such that $(a_i^{(\hat{k})} \cdot b_j^{(\hat{k})} + r_h) + (a_j^{(\hat{k})} \cdot b_i^{(\hat{k})} + r_l) \in \Omega$, or $(a_i^{(\hat{k})} \cdot b_j^{(\hat{k})} + r_h) \in \Omega$ (resp. $(a_j^{(\hat{k})} \cdot b_i^{(\hat{k})} + r_l) \in \Omega$) and $r_l \in \Omega$ (resp. $r_h \in \Omega$), then use the shares of the inputs with index in $I^{(\hat{k})}$ and $J^{(\hat{k})}$ and the same random bits simulated in \mathcal{R} ; otherwise, we can simulate the probe with a uniformly random value.

- For each probe in group (5), if $\exists \hat{k}$ such that $(a_j^{(\hat{k})} \cdot b_i^{(\hat{k})} + r_h) + (a_j^{(\hat{k})} \cdot b_i^{(\hat{k})} + r_l) \in \Omega$ or $c_i^{(\hat{k})} \in \Omega$, and $r_m \in \Omega$, then use the shares of the inputs with index in $I^{(k)}$ and $J^{(k)}$ and the same random bits r_h, r_l, r_m simulated at the beginning. Otherwise we can simulate the probe with a uniformly random value.

Since for each case the simulation uses only the shares of the inputs with indexes in $I^{(k)}, J^{(k)}$, and since $|I^{(k)}|, |J^{(k)}| \leq 3$, the simulation requires at most 3 shares per each input. Therefore $\text{Mult}_1^3, \dots, \text{Mult}_N^3$ is 3 – SCR. \square

In the following, we first present the proof of *weak t – SCR* for both orders 4 and 5, and then the ones for 4 – SNI and 5 – SNI. Due to the much higher number of possible probes, the security proofs for orders 4 and 5 will be more involved than the previous ones. Therefore after the proofs we will provide some insights on how the simulations work.

Proposition 3.10 (weak t – SCR for orders $t = 4, 5$). *Let $t = 4, 5$ and let $\text{Mult}_1^t, \dots, \text{Mult}_N^t$ be a set of N multiplication schemes as in Algorithm 3.4 for $t = 4$ or Algorithm 3.5 for $t = 5$. Let $(\mathbf{a}^{(1)}, \mathbf{b}^{(1)}), \dots, (\mathbf{a}^{(N)}, \mathbf{b}^{(N)})$ be the input values and $\mathbf{c}^{(1)}, \dots, \mathbf{c}^{(N)}$ the output values. Suppose that the maskings of the inputs are independent and uniformly chosen and that for $k = 1, \dots, N$ each Mult_k^t uses the same random bits $(r_i)_{i=1, \dots, tn/2}$. Then $\text{Mult}_1^t, \dots, \text{Mult}_N^t$ are weak t – SCR.*

Proof. Let $\Omega = (\mathcal{P}_1, \dots, \mathcal{P}_N)$ be a set of t observations respectively on $\text{Mult}_1^t, \dots, \text{Mult}_N^t$. We start by showing that $\text{Mult}_1^t, \dots, \text{Mult}_N^t$ satisfy Definition 3.3, i.e., that the probes in \mathcal{P}_k can be consistently simulated by at most $n - 1$ shares of the inputs of Mult_k^t , for each $k = 1, \dots, N$.

Probes classification.

Suppose p_1, \dots, p_t are t adversary's probes. We indicate with \mathbf{r}_{p_i} the vector of the respective randomness for every $i = 1, \dots, t$ and we classify such p_i in the following groups:

- (1) $\exists j \in [1, t]$ such that $\mathbf{r}_{p_i} = \mathbf{r}_{p_j}$
- (2) $\exists j \in [1, t]$ such that $\mathbf{r}_{p_i} \subset \mathbf{r}_{p_j}$
- (3) $\forall j \in [1, t] \mathbf{r}_{p_i} \cap \mathbf{r}_{p_j} = \emptyset$
- (4) $\mathbf{r}_{p_i} = \mathbf{0}$
- (5) $\exists \mathcal{J} \in [1, t]$ such that, for all $j \in \mathcal{J}$, $\mathbf{r}_{p_j} \subset \mathbf{r}_{p_i}$ and $\bigcup_j \mathbf{r}_{p_j} = \mathbf{r}_{p_i}$
- (6) $\forall j \in [1, t]$ such that $\mathbf{r}_{p_j} \subset \mathbf{r}_{p_i} : \bigcup_j \mathbf{r}_{p_j} \neq \mathbf{r}_{p_i}$

Construction of sets of shares indexes.

We define the sets $I_1, \dots, I_N, J_1, \dots, J_N$ with $|I_i| < n$ and $|J_i| < n$ for every $i = 1, \dots, N$ such that the values of the wires p_h can be perfectly simulated given the shares of the inputs $(a_i^{(1)})_{i \in I_1}, (b_i^{(1)})_{i \in J_1}, \dots, (a_i^{(N)})_{i \in I_N}, (b_i^{(N)})_{i \in J_N}$.

The procedure to construct the sets is the following:

- For every wire in the group (1), (2), (4) or (5), add all the indexes of the shares of $\mathbf{a}^{(j)}$ in I_j and of $\mathbf{b}^{(j)}$ in J_j for every $j = 1, \dots, N$.

We remark that

- by construction of Mult^t , for every probe in group (4) we add at most only one index to each set,
- there exist at most $\lfloor \frac{t}{2} \rfloor$ probes per each gadget in group (1),
- there exist at most $\lfloor \frac{t}{3} \rfloor$ probes per each gadget in group (5),
- for every probe p_i with $|\mathbf{r}_{p_i}| < t$, the number of indexes added is inferior then when $|\mathbf{r}_{p_i}| = t$.

Therefore, we have at most $\lfloor \frac{t}{2} \rfloor$ probes per gadget for which we add all the indexes of the shares in the sets I_j and J_j and thanks to the $\lfloor \frac{t}{2} \rfloor$ non completeness without randomness of Mult^t we can conclude that $|I_i| < n$ and $|J_i| < n$. In particular, we point out that if the inputs are not independent (or outputs of another $t - \text{SCR}$ and $t - \text{SNI}$ gadget), we cannot guarantee this bound on the sets of indexes, because the shares of inputs of a certain gadget might provide information regarding the shares of another one.

Simulation.

We now simulate, consistently with the randomness involved, the probes p_i by using $(a_i^{(1)})_{i \in I_1}, (b_i^{(1)})_{i \in J_1}, \dots, (a_i^{(N)})_{i \in I_N}, (b_i^{(N)})_{i \in J_N}$.

The simulation starts with a preliminary phase in which for every probe in group (1), (2) and (5) we pick at random the components of \mathbf{r}_{p_i} . Then, it follows the steps below, according to the different groups of probes.

- **Groups (1) and (2).** For every probe in group (1) and (2), we simulate p_i by using the random bits \mathbf{r}_{p_i} selected in the preliminary phase and the indexes of the inputs in $I_1, J_1, \dots, I_N, J_N$.
- **Group (3).** For every probe in group (3), since p_i does not share randomness with any other probe, we simulate it as a uniformly random bit.
- **Group (4).** For every probe in group (4), since p_i does not contain any randomness, we simulate it only by using the indexes of the inputs in $I_1, J_1, \dots, I_N, J_N$.
- **Group (5).** For every probe in group (5), since all the component of the randomness \mathbf{r}_{p_i} have been assigned in the preliminary phase, then we simulate p_i by using the random bits \mathbf{r}_{p_i} selected in the preliminary phase and the indexes of the inputs in I_1, J_1, I_2, J_2 .
- **Group (6).** For every probe in group (6), since p_i contains some randomness which does not appear elsewhere in the other probes, we simulate it as a uniformly random bit.

We showed that all possible sets of probes can be simulated using only the indexes of the input shares contained in I and J . Therefore, since $|I_i| < n$ and $|J_i| < n$, according to Definition 3.3 we conclude that $\text{Mult}_1^t, \dots, \text{Mult}_N^t$ are *weakly* $t - \text{SCR}$. \square

We provide now an intuition about the simulation above, by considering a representative attack to Mult^4 .

Let Mult_1^4 and Mult_2^4 be two multiplication schemes sharing the same randomness. As a toy example, we consider the critical situation when the adversary accesses the first two output shares of Mult_1^4 , let them be c_1, c_2 , and the first two output shares of Mult_2^4 , let them be c'_1, c'_2 :

$$\begin{aligned} c_1 &= r_{15} + a_1b_1 + r_1 + a_1b_2 + a_2b_1 + r_{18} + a_1b_3 + a_3b_1 + r_7 + a_2b_3 + a_3b_2 + r_{13} + r_8 \\ c_2 &= r_{16} + a_2b_2 + r_2 + a_2b_4 + a_4b_2 + r_{17} + a_2b_6 + a_6b_2 + r_1 + a_4b_6 + a_6b_4 + r_{14} \end{aligned}$$

$$\begin{aligned} c'_1 &= r_{15} + a'_1b'_1 + r_1 + a'_1b'_2 + a'_2b'_1 + r_{18} + a'_1b'_3 + a'_3b'_1 + r_7 + a'_2b'_3 + a'_3b'_2 + r_{13} + r_8 \\ c'_2 &= r_{16} + a'_2b'_2 + r_2 + a'_2b'_4 + a'_4b'_2 + r_{17} + a'_2b'_6 + a'_6b'_2 + r_1 + a'_4b'_6 + a'_6b'_4 + r_{14}. \end{aligned}$$

According to the classification of the probes in the proof above, since c_1, c_2 and c'_1, c'_2 share the same randomness they belong to group (1). We define the sets of indexes according to the procedure in the proof, by adding all the indexes of the input shares involved in the computation: $I := \{1, 2, 3, 4, 6\}$, $I' := \{1, 2, 3, 4, 6\}$, $J := \{1, 2, 3, 4, 6\}$, $J' := \{1, 2, 3, 4, 6\}$. The intuition behind this step is the fact that an adversary, by summing the probes up, could completely cancel out the randomness, and consequently discover all the input shares appearing in the computation. Thus the simulation needs to use such shares. Now, according to the property of $\lfloor \frac{t}{2} \rfloor$ – non-completeness, which Algorithm 3.4 satisfies by construction, each of these sets contains less than $n = 7$ indexes (in particular, in our example, they contain each 5 indexes). We simulate the probes using only the indexes in I, I', J, J' in the following way.

- First, we assign uniformly at random all the random bits involved in the computation, i.e., $r_1, r_2, r_{13}, r_{18}, r_{17}, r_7, r_{16}, r_{14}, r_{15}$.
- The, we compute c_1, c_2 and c'_1, c'_2 exactly as in the real algorithm, by using the shares a_i with $i \in I$, b_j with $j \in J$, a'_i with $i \in I'$ and b'_j with $j \in J'$.

Since the simulation uses only input shares with index in I, J, I', J' , which have cardinality 5, the conditions in Definition 3.3 are satisfied and the gadgets are *weakly 4 – SCR*.

Proposition 3.11 (t – SNI for order $t = 4$). *Let Mult^4 be a multiplication scheme as in Algorithm 3.4, with inputs \mathbf{a}, \mathbf{b} and output \mathbf{c} . Then Mult^4 is 4 – SNI.*

Proof. Let $\Omega = (\mathcal{I}, \mathcal{O})$ be a set of 4 observations respectively on the internal and on the output wires, where $|\mathcal{I}| = t_1$ and in particular $t_1 + |\mathcal{O}| \leq 4$.

Probes classification.

We classify the internal wires in the following groups:

- (1) $a_i, b_j, a_i b_j$
- (2.1) r_k
- (2.2) $r_k + a_i b_i$
- (3.1) $r_k + a_i b_i + r_h,$
- (3.2) $r_k + a_i b_i + r_h + a_i b_j,$
- (3.3) $r_k + a_i b_i + r_h + a_i b_j + a_j b_i,$
- (4.1) $r_k + a_i b_i + r_h + a_i b_j + a_j b_i + r_l,$
- (4.2) $r_k + a_i b_i + r_h + a_i b_j + a_j b_i + r_l + a_i b_u,$
- (4.3) $r_k + a_i b_i + r_h + a_i b_j + a_j b_i + r_l + a_i b_u + a_u b_i,$
- (5.1) $r_k + a_i b_i + r_h + a_i b_j + a_j b_i + r_l + a_i b_u + a_u b_i + r_m,$
- (5.2) $r_k + a_i b_i + r_h + a_i b_j + a_j b_i + r_l + a_i b_u + a_u b_i + r_m + a_j b_u,$
- (5.3) $r_k + a_i b_i + r_h + a_i b_j + a_j b_i + r_l + a_i b_u + a_u b_i + r_m + a_j b_u + a_u b_j$
- (6) $c_i = r_k + a_i b_i + r_h + a_i b_j + a_j b_i + r_l + a_i b_u + a_u b_i + r_m + a_j b_u + a_u b_j + r_e$
- (7) $c_1 = r_k + a_i b_i + r_h + a_i b_j + a_j b_i + r_l + a_i b_u + a_u b_i + r_m + a_j b_u + a_u b_j + r_e + r_f$

Construction of sets of shares indexes.

Suppose an adversary corrupts at most 4 wires w_1, \dots, w_4 . We define two sets I, J with $|I| < t_1$ and $|J| < t_1$ such that the values of the wires w_h can be perfectly simulated given the values $(a_i)_{i \in I}, (b_i)_{i \in J}$.

The procedure to construct the sets is the following. We first define a set K such that for all the probes in group (2.1) and (2.2) we add k to K . Initially I, J are empty and the w_i unassigned. Then

- **Group (1).** For every wire in the group (1) or a combination of this, add i to I and j to J .
- **Group (2.2).** For every wire in group (2.2) if $i \notin I$, add i to I , if $i \notin J$, add i to J .
- **Group (3.1).** For every wire in group (3.1) and such that $k, h \in K$, if $i \notin I$, for every index of the shares of a which is not in I , add the index to I and for every index of the shares of b which is not in J , add the index to J .
- **Group (3.2).**
 - For every wire in group (3.2) and such that $k, h \in K$, if $i \notin I$, for every index of the shares of a which is not in I , add the index to I and for every index of the shares of b which is not in J , add the index to J .
 - * For every wire in group (3.2) such that $r_k + a_i b_i + r_h$ was probed, if $i \notin I$, add i to I , if $j \notin I$, add j to I , if $i \notin J$, add i to J , if $j \notin J$, add j to J .
 - * For every wire in group (3.2) or (3.3) such that $r_k + a_i b_i$ was probed and $h \in K$, if $i \notin I$, add i to I , if $j \notin I$, add j to I , if $i \notin J$, add i to J , if $j \notin J$, add j to J .
- **Group (3.3).**
 - For every wire in group (3.1), (3.2), (3.3) and such that $k, h \in K$, if $i \notin I$, for every index of the shares of a which is not in I , add the index to I and for every

- index of the shares of b which is not in J , add the index to J .
- * For every wire in group (3.3) such that $r_k + a_i b_i + r_h$ was probed, if $i \notin I$, add i to I , if $j \notin I$, add j to I , if $i \notin J$, add i to J , if $j \notin J$, add j to J .
 - * For every wire in group (3.3) such that $r_k + a_i b_i + r_h + a_i b_j$ was probed if $j \notin I$, add j to I , if $i \notin J$, add i to J .
- **Group (4.1).**
 - For every wire in group (4.1) and such that $k \in K$, $h \in K$ and $l \in K$, for every index of the shares of a which is not in I , add the index to I and for every index of the shares of b which is not in J , add the index to J .
 - For every wire in group (4.1) such that $r_k + a_i b_i + r_h$ was probed and $l \in K$, or such that $r_k + a_i b_i$ was probed and $h, l \in K$, if $i \notin I$, add i to I , if $i \notin J$, add i to J , if $j \notin I$, add j to I , if $j \notin J$, add j to J .
 - **Group (4.2).**
 - For every wire in group (4.2) and such that $k \in K$, $h \in K$ and $l \in K$, for every index of the shares of a which is not in I , add the index to I and for every index of the shares of b which is not in J , add the index to J .
 - For every wire in group (4.2) such that $r_k + a_i b_i + r_h$ was probed and $l \in K$, if $i \notin I$, add i to I , if $i \notin J$, add i to J , if $u \notin I$, add u to I , if $u \notin J$, add u to J .
 - **Group (4.3).**
 - For every wire in group (4.3) and such that $k \in K$, $h \in K$ and $l \in K$, for every index of the shares of a which is not in I , add the index to I and for every index of the shares of b which is not in J , add the index to J .
 - * For every wire in group (4.3) such that $r_k + a_i b_i + r_h + a_i b_j + a_j b_i + r_l$ was probed, if $i \notin I$, add i to I , if $i \notin J$, add i to J , if $u \notin I$, add u to I , if $u \notin J$, add u to J .
 - For every wire in group (4.3) such that $r_k + a_i b_i + r_h$ was probed and $l \in K$, if $i \notin I$, add i to I , if $i \notin J$, add i to J , if $u \notin I$, add u to I , if $u \notin J$, add u to J .
 - **Group (5.1).**
 - For every wire in group (5.1) such that $r_k + a_i b_i + r_h$ was probed and $l, m \in K$, if $i \notin I$, add i to I , if $i \notin J$, add i to J , if $j \notin I$, add j to I , if $j \notin J$, add j to J , if $u \notin I$, add u to I , if $u \notin J$, add u to J .
 - For every wire in group (5.1) such that $r_k + a_i b_i + r_h + a_i b_j + a_j b_i + r_l$ was probed and $m \in K$, if $i \notin I$, add i to I , if $i \notin J$, add i to J , if $j \notin I$, if $u \notin I$, add u to I , if $u \notin J$, add u to J .
 - **Group (5.2).**
 - For every wire in group (5.2) such that $r_k + a_i b_i + r_h + a_i b_j + a_j b_i + r_l + a_i b_u +$

- $a_u b_i + r_m$ was probed and $m \notin K$, if $j \notin I$, add j to I , if $j \notin J$, add j to J , if $u \notin I$, add u to I , if $u \notin J$, add u to J .
- For every wire in group (5.2) such that $r_k + a_i b_i + r_h + a_i b_j + a_j b_i + r_l$ was probed and $m \in K$, if $i \notin I$, add i to I , if $i \notin J$, add i to J , if $j \notin I$, add j to I , if $j \notin J$, add j to J , if $u \notin I$, add u to I , if $u \notin J$, add u to J .
- For every wire in group (5.2) such that $r_k + a_i b_i + r_h$ was probed and $l, m \in K$, if $i \notin I$, add i to I , if $i \notin J$, add i to J , if $j \notin I$, add j to I , if $j \notin J$, add j to J , if $u \notin I$, add u to I , if $u \notin J$, add u to J .
- **Group (5.3).**
 - For every wire in group (5.3) such that $r_k + a_i b_i + r_h + a_i b_j + a_j b_i + r_l + a_i b_u + a_u b_i + r_m$ was probed and $m \notin K$, if $j \notin I$, add j to I , if $j \notin J$, add j to J , if $u \notin I$, add u to I , if $u \notin J$, add u to J .
 - For every wire in group (5.3) such that $r_k + a_i b_i + r_h + a_i b_j + a_j b_i + r_l$ was probed and $m \in K$, if $i \notin I$, add i to I , if $i \notin J$, add i to J , if $j \notin I$, add j to I , if $j \notin J$, add j to J , if $u \notin I$, add u to I , if $u \notin J$, add u to J .
 - For every wire in group (5.3) such that $r_k + a_i b_i + r_h$ was probed and $l, m \in K$, if $i \notin I$, add i to I , if $i \notin J$, add i to J , if $j \notin I$, add j to I , if $j \notin J$, add j to J , if $u \notin I$, add u to I , if $u \notin J$, add u to J .

We now point out that $|I| < t_1$ and $|J| < t_1$. Indeed, if $t_1 = 1$, according to the distribution of the randomness, the only groups which add indexes to I and J are groups (1) and (2.2). Therefore $|I| \leq 1$ and $|J| \leq 1$. If $t_1 = 2$, then the only groups which add indexes to I and J can be the ones marked with *. Therefore, in all the previous situations $|I| \leq 2$ and $|J| \leq 2$. If $t_1 \geq 3$, then the groups which add indexes to I and J are all the remaining and in any of these cases $|I| \leq 3$ and $|J| \leq 3$ if $t_1 = 3$ or $|I| \leq 4$ and $|J| \leq 4$ if $t_1 = 4$.

Simulation.

We now simulate the wires w_1, \dots, w_4 using only the input shares $(a_i)_{i \in I}$ and $(b_j)_{j \in J}$. Initially, for every $k \in K$ assign r_k to a random and independent value. This simulates the probes in (2.1). Then we follow the procedure below.

- **Group (1).** For every probe in group (1), then by construction $i \in I$ and $j \in J$ and the values are perfectly simulated.
- **Group (2.2).** For every probe in group (2.2), r_k has been simulated in the first step of the simulation. Moreover, by construction $i \in I$ and $j \in J$ and therefore the probe can be perfectly simulated.
- **Group (3.1).** For every probe in (3.1)
 - if r_k and r_h have been observed, then the values have been assigned to a random and independent value at the beginning of the simulation and, since by construction $i \in I, J$, the probe can be simulated by using r_k, r_h and the required shares of a and b ;

- otherwise, if r_h or r_k has not been observed, then the probe can be simulated as a uniformly random value.
- **Group (3.2).** For every probe in (3.2)
 - if r_k and r_h have been observed, then the values have been assigned to a random and independent value at the beginning of the simulation and, since by construction $i, j \in I, J$, the probe can be simulated by using r_k, r_h and the required shares of a and b ;
 - otherwise, if r_h or r_k has not been observed, then the probe can be simulated as a uniformly random value.
 - If $p := r_k + a_i b_i + r_h$ was probed, then by construction $i, j \in I, J$ and the probe can be simulated by summing p and $a_i b_j$;
 - otherwise, if $p := r_k + a_i b_i$ and r_h have been probed, then r_h has been assigned to a random and independent value at the beginning of the simulation and by construction $i, j \in I, J$. Therefore, the probe can be simulated by summing p, r_h and the needed shares of a and b .
 - Finally, if $p := r_k + a_i b_i$ but r_h has not been probed, then the observation can be simulated as a random and independent value.
- **Group (3.3).** For every probe in (3.3), the simulation proceeds in the same way as in the previous group. Moreover, if the value $p := r_k + a_i b_i + r_h + a_i b_j$ has been probed, then by construction $i \in J$ and $j \in I$, therefore the probe can be simulated by adding p and $a_j b_i$.
- **Group (4.1).** For every probe in group (4.1)
 - if r_k or $r_k + a_i b_i$ and r_h and r_l have been observed, then the values of the random bits have been assigned to a random and independent value at the beginning of the simulation and, since by construction $i, j \in I, J$, the probe can be simulated by using r_k, r_h, r_l and the required shares of a and b ;
 - if r_h or r_k or r_l has not been observed, then the probe can be simulated as a uniformly random value;
 - if $p := r_k + a_i b_i + r_h$ and r_l have been probed, then r_l has been assigned to a random and independent value at the beginning of the simulation and by construction $i, j \in I, J$. Therefore the probe can be simulated by using p, r_l and a_i, a_j, b_i, b_j ;
 - otherwise we can simulate the probe as a random and independent value.
- **Group (4.2).** For every probe in group (4.2)
 - if r_k or $r_k + a_i b_i$ and r_h and r_l have been observed, then the values of the random bits have been assigned to a random and independent value at the beginning of the simulation and, since by construction $i, j, u \in I, J$, the probe can be simulated by using r_k, r_h, r_l and the required shares of a and b .

- if $p := r_k + a_i b_i + r_h$ and r_l have been probed, then r_l has been assigned to a random and independent value at the beginning of the simulation and by construction $i, j, u \in I, J$. Therefore the probe can be simulated by using p , r_l and a_i, a_j, b_i, b_j, b_u ;
- if $p := r_k + a_i b_i + r_h + a_i b_j + a_j b_i + r_l$ has been probed, then by construction $i \in I$ and $u \in J$. Therefore the probe can be simulated by using p and a_i, b_u ;
- otherwise we can simulate the probe as a random and independent value.
- **Group (4.3).** For every probe in group (4.3)
 - if r_k or $r_k + a_i b_i$ and r_h and r_l have been observed, then the values of the random bits have been assigned to a random and independent value at the beginning of the simulation and, since by construction $i, j \in I, J$, the probe can be simulated by using r_k, r_h, r_l and the required shares of a and b ;
 - if $p := r_k + a_i b_i + r_h$ and r_l have been probed, then r_l has been assigned to a random and independent value at the beginning of the simulation and by construction $i, j, u \in I, J$. Therefore the probe can be simulated by using p , r_l and $a_i, a_j, a_u, b_i, b_j, b_u$;
 - if $p := r_k + a_i b_i + r_h + a_i b_j + a_j b_i + r_l$ has been probed then by construction the indexes $i, u \in I, J$. Therefore the probe can be simulated by using p and a_i, a_u, b_i, b_u ;
 - otherwise we can simulate the probe as a random and independent value.
- **Groups (5.1), (5.2) and (5.3).** For every probe in group (5.1), (5.2) or (5.3)
 - if $p := r_k + a_i b_i + r_h$ and r_l, r_m have been probed, then r_l and r_m have been assigned to a random and independent value at the beginning of the simulation and by construction $i, j, u \in I, J$. Therefore the probe can be simulated by using p , r_l , r_m and $a_i, a_j, a_u, b_i, b_j, b_u$;
 - if $p := r_k + a_i b_i + r_h + a_i b_j + a_j b_i + r_l$ and r_m have been probed, then by construction $i, u \in I, J$. Therefore the probe can be simulated by using p and a_i, a_u, b_i, b_u ;
 - otherwise we can simulate the probe as a random and independent value.
- **Groups (6) and (7).** For every probe in group (6) or (7), since there exists a random bit which was not probed, the value can be simulated as a random and independent value.

We just showed that all possible sets of probes can be simulated using only the indexes of the input shares contained in I and J . Therefore, since $|I| < t_1$ and $|J| < t_1$, according to Definition 2.5 we conclude that Mult^4 is 4 – SNI. \square

Again, we give an intuition on the proof of t – SNI by showing an exemplary attack. Let us suppose that an attacker has two internal probes p_1, p_2 and two external probes c_1, c_2 ,

looking as in the following

$$p_1 = r_{18}$$

$$p_2 = r_{18} + a_5b_5 + r_9 + a_5b_2 + a_2b_5$$

$$c_1 = r_{15} + a_1b_1 + r_1 + a_1b_2 + a_2b_1 + r_{18} + a_1b_3 + a_3b_1 + r_7 + a_2b_3 + a_3b_2 + r_{13} + r_8$$

$$c_2 = r_{16} + a_2b_2 + r_2 + a_2b_4 + a_4b_2 + r_{17} + a_2b_6 + a_6b_2 + r_1 + a_4b_6 + a_6b_4 + r_{14}$$

According to the probes classification in the proof, p_1 is in group (2.1), and therefore, as a preliminary phase, we need to add r_{18} to a set K . Then, we define sets of indexes, I and J , as in the following. Since p_2 is in group (3.3), $r_{18} \in K$ but $r_9 \notin K$, and none of its partial addends was probed, we do not add any index to I or J . The insight at the base of this is the fact that, thanks to the random bits r_{18} and r_9 , which cannot be canceled out with any of the other probes, p_2 looks to the adversary uniformly random and therefore no indexes are needed for its simulation. Finally, since c_1 is in group (6) and c_2 in group (7), in both cases we do not need to add any index to I or J . Indeed, since they differ to each other and to p_1 and p_2 of at least one random bit, they look to the adversary uniformly random. At this point, the simulation phase is quick. Thanks to the arguments above, we simulate uniformly at random the probes p_2 , c_1 and c_3 and we assign p_1 to r_{18} .

Similar principles are at the base of the proof of 5 – SNI.

Proposition 3.12 (t – SNI for order $t = 5$). *Let Mult^5 be a multiplication scheme as in Algorithm 3.5, with inputs \mathbf{a}, \mathbf{b} and output \mathbf{c} . Then Mult^5 is 5 – SNI.*

Proof. Let $\Omega = (\mathcal{I}, \mathcal{O})$ be a set of 5 observations respectively on the internal and on the output wires, where $|\mathcal{I}| = t_1$ and $|\mathcal{O}| = t_2$ (and in particular $t_1 + t_2 \leq 5$).

Probes classification.

We can classify the internal wires in the following groups:

- (1) $a_i, b_j, a_i b_j$
- (2) $a_i b_j + r_k$
- (3) r_k
- (4) $c_{i,j}$, which corresponds to a partial sum of the output share c_i .

For example, $c_{1,0} = a_1 b_1$, $c_{1,2} = a_1 b_1 + (a_1 b_2 + r_{19}) + (a_2 b_1 + r_{15})$.

Construction of sets of shares indexes.

Suppose an adversary probes at most t wires w_1, \dots, w_t . We define two sets I, J with $|I| < t_1$ $|J| < t_1$ such that the values of the wires w_h can be perfectly simulated

given the values $(a_i)_{i \in I}, (b_i)_{i \in J}$.

The procedure to construct the sets is the following:

1. We first define a set K such that for all probes in group (2), (3) or (4) we add the index k of each random bit r_k to K .
2. Initially I, J are empty and the w_i unassigned.
3. For every wire in group (1) or a combination of this, if $i \notin I$ add i to I and if $j \notin J$ add j to J .
4. For every wire in group (2), add i to I and j to J .
5. For every wire in group (4) such that all the indexes of the random bits appear in K at least two times, for every share a_i observed, if $i \notin I$, add i to I and for every share b_j observed, if $j \notin J$, add j to J .
6. For every wire in group (4) such that a certain $c_{i,j-h}$ was probed for $h=1, \dots, j-1$, if for every r_k in $c_{i,j-h} - c_{i,j-h} - k \in K$, then add all the indexes of the shares of a in I and all the shares of b in J , unless they are already in I and J .

We point out that the sets just constructed are such that $|I| \leq t_1$ and $|J| \leq t_1$. Indeed, for each probe in group (1) and (2) at most one index is added. For each probe in (4), we add more than one index to I and J only in the following four possible scenarios:

(a) The attacker's probes are of the form

$$\begin{aligned} - p_1 &:= (a_i b_j + r_{k_1}) + (a_e b_f + r_{k_2}) \\ - p_2 &:= a_g b_h + r_{k_1} \\ - p_3 &:= a_l b_m + r_{k_2} \end{aligned}$$

(b) The attacker's probes are of the form

$$\begin{aligned} - p_1 &:= (a_i b_j + r_{k_1}) + (a_e b_f + r_{k_2}) + (a_p b_q + r_{k_3}) \\ - p_2 &:= a_g b_h + r_{k_1} \\ - p_3 &:= a_l b_m + r_{k_2} \\ - p_4 &:= a_r b_s + r_{k_3} \end{aligned}$$

(c) The attacker's probes are of the form

$$\begin{aligned} - p_1 &:= (a_i b_j + r_{k_1}) + (a_e b_f + r_{k_2}) + (a_p b_q + r_{k_3}) + (a_u b_v + r_{k_4}) \\ - p_2 &:= a_g b_h + r_{k_1} \\ - p_3 &:= a_l b_m + r_{k_2} \\ - p_4 &:= a_r b_s + r_{k_3} \\ - p_5 &:= a_w b_z + r_{k_4} \end{aligned}$$

(d) The attacker's probes are of the form

$$\begin{aligned} - p_1 &:= (a_i b_j + r_{k_1}) + (a_e b_f + r_{k_2}) + (a_p b_q + r_{k_3}) \\ - p_2 &:= (a_g b_h + r_{k_1}) + (a_u b_v + r_{k_4}) \\ - p_3 &:= a_l b_m + r_{k_2} \\ - p_4 &:= a_r b_s + r_{k_3} \\ - p_5 &:= a_w b_z + r_{k_4} \end{aligned}$$

We give below a table with all the possible sets of probes in the previous four scenarios and the corresponding cardinality of I and J , representing the number of shares needed for the simulation. We can see that it always holds $|I| \leq t_1$ and $|J| \leq t_1$.

Probes in scenario (a)	$ I $	$ J $
$p_1 := (a_2b_3 + r_1) + (a_3b_2 + r_7), p_2 := (a_2b_4 + r_1), p_3 := (a_3b_4 + r_7)$	2	3
$p_1 := (a_4b_2 + r_2) + (a_2b_4 + r_1), p_2 := (a_2b_3 + r_1), p_3 := (a_1b_4 + r_2)$	3	3
$p_1 := (a_5b_4 + r_3) + (a_1b_4 + r_2), p_2 := (a_4b_2 + r_2), p_3 := (a_5b_7 + r_3)$	3	3
$p_1 := (a_2b_7 + r_4) + (a_5b_7 + r_3), p_2 := (a_7b_6 + r_4), p_3 := (a_5b_4 + r_3)$	3	3
$p_1 := (a_6b_7 + r_5) + (a_7b_6 + r_4), p_2 := (a_6b_3 + r_5), p_3 := (a_2b_7 + r_4)$	3	3
$p_1 := (a_3b_6 + r_6) + (a_6b_3 + r_5), p_2 := (a_3b_7 + r_6), p_3 := (a_6b_7 + r_5)$	2	3
$p_1 := (a_3b_4 + r_7) + (a_7b_4 + r_{14}), p_2 := (a_3b_2 + r_7), p_3 := (a_4b_6 + r_{14})$	2	3
Probes in scenario (b)		
$p_1 := (a_2b_3 + r_1) + (a_3b_2 + r_7) + (a_1b_3 + r_8), p_2 := (a_2b_4 + r_1),$ $p_3 := (a_3b_4 + r_7), p_4 := (a_1b_5 + r_8)$	3	4
$p_1 := (a_4b_2 + r_2) + (a_2b_4 + r_1) + (a_6b_2 + r_9), p_2 := (a_2b_3 + r_1),$ $p_3 := (a_1b_4 + r_2), p_4 := (a_2b_5 + r_9)$	4	4
$p_1 := (a_5b_4 + r_3) + (a_1b_4 + r_2) + (a_4b_5 + r_{10}), p_2 := (a_4b_2 + r_2),$ $p_3 := (a_5b_7 + r_3), p_4 := (a_1b_7 + r_{10})$	3	4
$p_1 := (a_2b_7 + r_4) + (a_5b_7 + r_3) + (a_5b_2 + r_{11}), p_2 := (a_7b_6 + r_4),$ $p_3 := (a_5b_4 + r_3), p_4 := (a_5b_6 + r_{11})$	3	4
$p_1 := (a_6b_7 + r_5) + (a_7b_6 + r_4) + (a_7b_1 + r_{12}), p_2 := (a_6b_3 + r_5),$ $p_3 := (a_2b_7 + r_4), p_4 := (a_4b_7 + r_{12})$	4	4
$p_1 := (a_3b_6 + r_6) + (a_6b_3 + r_5) + (a_5b_3 + r_{13}), p_2 := (a_3b_7 + r_6),$ $p_3 := (a_6b_7 + r_5), p_4 := (a_3b_1 + r_{13})$	3	1
$p_1 := (a_3b_4 + r_7) + (a_7b_4 + r_{14}) + (a_3b_7 + r_6), p_2 := (a_3b_2 + r_7),$ $p_3 := (a_4b_6 + r_{14}), p_4 := (a_3b_6 + r_6)$	3	4
Probes in scenario (c)		
$p_1 := (a_2b_3 + r_1) + (a_3b_2 + r_7) + (a_1b_3 + r_8) + (a_3b_1 + r_{13}),$ $p_2 := (a_2b_4 + r_1), p_3 := (a_3b_4 + r_7), p_4 := (a_1b_5 + r_8), p_5 := (a_5b_3 + r_{13})$	4	5
$p_1 := (a_4b_2 + r_2) + (a_2b_4 + r_1) + (a_6b_2 + r_9) + (a_4b_6 + r_{14}),$ $p_2 := (a_2b_3 + r_1), p_3 := (a_1b_4 + r_2), p_4 := (a_2b_5 + r_9), p_5 := (a_7b_4 + r_{14})$	5	5

3. Randomness optimization in Boolean masking

$p_1 := (a_5b_4 + r_3) + (a_1b_4 + r_2) + (a_4b_5 + r_{10}) + (a_1b_5 + r_8),$ $p_2 := (a_4b_2 + r_2), p_3 := (a_5b_7 + r_3), p_4 := (a_1b_7 + r_{10}), p_5 := (a_1b_3 + r_8)$	3	5
$p_1 := (a_2b_7 + r_4) + (a_5b_7 + r_3) + (a_5b_2 + r_{11}) + (a_2b_5 + r_9),$ $p_2 := (a_7b_6 + r_4), p_3 := (a_5b_4 + r_3), p_4 := (a_5b_6 + r_{11}), p_5 :=$	3	5
$p_1 := (a_6b_7 + r_5) + (a_7b_6 + r_4) + (a_7b_1 + r_{12}) + (a_1b_7 + r_{10}),$ $p_2 := (a_6b_3 + r_5), p_3 := (a_2b_7 + r_4), p_4 := (a_4b_7 + r_{12}), p_5 := (a_6b_2 + r_9)$	5	5
$p_1 := (a_3b_6 + r_6) + (a_6b_3 + r_5) + (a_5b_3 + r_{13}) + (a_5b_6 + r_{11}),$ $p_2 := (a_3b_7 + r_6), p_3 := (a_6b_7 + r_5), p_4 := (a_3b_1 + r_{13}), p_5 := (a_5b_2 + r_{11})$	3	5
$p_1 := (a_3b_4 + r_7) + (a_7b_4 + r_{14}) + (a_3b_7 + r_6) + (a_4b_7 + r_{12}),$ $p_2 := (a_3b_2 + r_7), p_3 := (a_4b_6 + r_{14}), p_4 := (a_3b_6 + r_6), p_5 := (a_7b_1 + r_{12})$	4	5
Probes in scenario (d)		
$p_1 := (a_2b_3 + r_1) + (a_3b_2 + r_7) + (a_1b_3 + r_8), p_2 := (a_3b_4 + r_7),$ $p_3 := (a_4b_2 + r_2) + (a_2b_4 + r_1), p_4 := (a_1b_5 + r_8), p_5 := (a_1b_4 + r_2)$	4	4
$p_1 := (a_2b_3 + r_1) + (a_3b_2 + r_7) + (a_1b_3 + r_8), p_2 := (a_2b_4 + r_1),$ $p_3 := (a_3b_4 + r_7) + (a_7b_4 + r_{14}), p_4 := (a_1b_5 + r_8), p_5 := (a_7b_4 + r_{14})$	5	5
$p_1 := (a_4b_2 + r_2) + (a_2b_4 + r_1) + (a_6b_2 + r_9), p_2 := (a_2b_5 + r_9),$ $p_3 := (a_2b_3 + r_1) + (a_3b_2 + r_7), p_4 := (a_3b_4 + r_7), p_5 := (a_1b_4 + r_2)$	5	3
$p_1 := (a_4b_2 + r_2) + (a_2b_4 + r_1) + (a_6b_2 + r_9), p_2 := (a_2b_5 + r_9),$ $p_3 := (a_5b_4 + r_3) + (a_1b_4 + r_2), p_4 := (a_2b_3 + r_1), p_5 := (a_5b_7 + r_3)$	5	5
$p_1 := (a_5b_4 + r_3) + (a_1b_4 + r_2) + (a_4b_5 + r_{10}), p_2 := (a_1b_7 + r_{10}),$ $p_3 := (a_4b_2 + r_2) + (a_2b_4 + r_1), p_4 := (a_5b_7 + r_3), p_5 := (a_2b_3 + r_1)$	4	5
$p_1 := (a_2b_7 + r_4) + (a_5b_7 + r_3) + (a_5b_2 + r_{11}), p_2 := (a_7b_6 + r_4),$ $p_3 := (a_5b_4 + r_3) + (a_1b_4 + r_2), p_4 := (a_4b_2 + r_2), p_5 := (a_5b_6 + r_{11})$	5	4
$p_1 := (a_2b_7 + r_4) + (a_5b_7 + r_3) + (a_5b_2 + r_{11}), p_2 := (a_5b_4 + r_3),$ $p_3 := (a_6b_7 + r_5) + (a_7b_6 + r_4), p_4 := (a_6b_3 + r_5), p_5 := (a_5b_6 + r_{11})$	4	5
$p_1 := (a_6b_7 + r_5) + (a_7b_6 + r_4) + (a_7b_1 + r_{12}), p_2 := (a_5b_4 + r_3),$ $p_3 := (a_2b_7 + r_4) + (a_5b_7 + r_3), p_4 := (a_6b_3 + r_5), p_5 := (a_4b_7 + r_{12})$	5	5
$p_1 := (a_3b_6 + r_6) + (a_6b_3 + r_5) + (a_5b_3 + r_{13}), p_2 := (a_2b_7 + r_4),$ $p_3 := (a_6b_7 + r_5) + (a_7b_6 + r_4), p_4 := (a_3b_1 + r_{13}), p_5 := (a_3b_7 + r_6)$	5	5
$p_1 := (a_3b_4 + r_7) + (a_7b_4 + r_{14}) + (a_3b_7 + r_6), p_2 := (a_2b_4 + r_1),$ $p_3 := (a_2b_3 + r_1) + (a_3b_2 + r_7), p_4 := (a_3b_6 + r_6), p_5 := (a_4b_6 + r_{14})$	4	5
$p_1 := (a_3b_4 + r_7) + (a_7b_4 + r_{14}) + (a_3b_7 + r_6), p_2 := (a_6b_7 + r_5),$ $p_3 := (a_3b_6 + r_6) + (a_6b_3 + r_5), p_4 := (a_3b_2 + r_7), p_5 := (a_4b_6 + r_{14})$	4	5

Simulation.

Now we simulate the wires w_h using only the values $(a_i)_{i \in I}$ and $(b_i)_{i \in J}$.

- We start the simulation with a preliminary phase in which for every $k \in K$ we assign r_k to a random and independent value.
- For every probe in group (1), $i \in I$ and $i \in J$. Therefore the values are perfectly simulated.
- For every probe in group (2), since $k \in K$ and r_k has been simulated in the preliminary phase as a random and independent value, we can compute w_i by using r_k and the indexes of the inputs which are, by construction, in the sets I and J .
- For every probe in group (4):
 - if for all the random bits r_k used in the computation, $k \in K$, then by construction the indexes of the shares of a and b are respectively in I and J and the bits r_k have been taken randomly in the initial step of the simulation. Therefore the probe can be simulated using these values.
 - if none of the sums $c_{i,j-h}$ have been probed and there exists at least one random bit r_k used in the computation such that $k \notin K$, then $c_{i,j}$ can be simulated as a uniform and random value.
 - if a sum $c_{i,j-h}$, with $h = j-1, \dots, 1$, has been probed and every r_k in $c_{i,j-h}$ is such that $k \in K$, then we can simulate $c_{i,j}$ from r_k assigned at random in the preliminary phase, the probe $c_{i,j-h}$ and the shares of a and b which are in I and J by construction.
 - if a sum $c_{i,j-h}$, with $h = j-1, \dots, 1$, has been probed and there exists a random bit r_k in $c_{i,j} - c_{i,j-h}$ such that $k \notin K$, then we can simulate $c_{i,j}$ as a uniform and random bit.

Finally, the simulation of the output probes c_i follows the same steps of the probes in group (4) and it can be therefore performed by using at most t_1 shares of the inputs, completing the proof. \square

3.3. A t – SCR refreshing scheme

In the following, we show that the refreshing scheme in Algorithm 2.3 is t – SCR. We remark that, due to the use of $n > t + 1$ shares in the multiplication algorithm for order $t > 3$, the refreshing scheme in Algorithm 2.3 makes use of a significant amount of randomness, more precisely $\frac{n(n-1)}{2}$ random bits, which can be optimized. In the following we propose a more efficient t – SCR refreshing scheme for orders $t \geq 4$. The principle at the base of such a refreshing is to add fresh random bits to each input share such that there are t random bits involved in the computation of each output share and each of such random bits is used a second time in a distinct output share. The new scheme makes use of $\frac{n-t}{2}$ random bits. An example for order $t = 4$ is depicted in

Algorithm 3.6 Refreshing scheme \mathcal{R}' for order $t = 4$

Input: shares a_1, \dots, a_7 such that $\bigoplus a_i = a$

Output: shares c_1, \dots, c_7 such that $\bigoplus c_i = a$

- 1: $c_1 = a_1 + r_1 + r_7 + r_{13} + r_8;$
 - 2: $c_2 = a_2 + r_2 + r_1 + r_{14} + r_9;$
 - 3: $c_3 = a_3 + r_8 + r_{10} + r_3 + r_2;$
 - 4: $c_4 = a_4 + r_9 + r_4 + r_3 + r_{11};$
 - 5: $c_5 = a_5 + r_5 + r_{10} + r_4 + r_{12};$
 - 6: $c_6 = a_6 + r_6 + r_{13} + r_{11} + r_5;$
 - 7: $c_7 = a_7 + r_{14} + r_6 + r_{12} + r_7;$
-

Algorithm 3.6. In Lemma 3.13 we show that both the refreshing schemes just mentioned are $t - \text{SCR}$. In particular, we point out that the proof is general and applies not only to Algorithm 3.6, but to any refreshing scheme at any other order built according to the description aforementioned. In Lemma 3.14 we show that the new refreshing scheme is $t - \text{SNI}$.

Lemma 3.13. *Let $\mathcal{R}'_1, \dots, \mathcal{R}'_N$ be a set of N multiplication schemes as in Algorithm 2.3 or Algorithm 3.6, with inputs $\mathbf{a}^{(1)}, \dots, \mathbf{a}^{(N)}$ and outputs $\mathbf{c}^{(1)}, \dots, \mathbf{c}^{(N)}$. Suppose that the maskings of the inputs are independent and uniformly chosen, and that for $k = 1, \dots, N$ each \mathcal{R}'_k uses the same random bits $(r_i)_{i=1, \dots, tn/2}$. Then $\mathcal{R}'_1, \dots, \mathcal{R}'_N$ are $t - \text{SCR}$.*

Proof. Let $\Omega = (\mathcal{P}_1, \dots, \mathcal{P}_N)$ be a set of t observations respectively on $\mathcal{R}'_1, \dots, \mathcal{R}'_N$. We show that $\mathcal{R}'_1, \dots, \mathcal{R}'_N$ satisfy Definition 3.4, i.e., that the probes in \mathcal{P}_k can be consistently simulated by at most $n - 1$ shares of the inputs of \mathcal{R}'_k , for each $k = 1, \dots, N$.

Suppose p_1, \dots, p_t are t adversary's probes. We indicate with \mathbf{r}_{p_i} the vector of the respective randomness for every $i = 1, \dots, t$ and we classify such p_i in the following groups:

- (1) $\exists j \in [1, t]$ such that $\mathbf{r}_{p_i} = \mathbf{r}_{p_j}$
- (2) $\exists j \in [1, t]$ such that $\mathbf{r}_{p_i} \subset \mathbf{r}_{p_j}$
- (3) $\forall j \in [1, t] \mathbf{r}_{p_i} \cap \mathbf{r}_{p_j} = \emptyset$
- (4) $\mathbf{r}_{p_i} = \mathbf{0}$
- (5) $\exists j \in [1, t]$ such that $\mathbf{r}_{p_j} \subset \mathbf{r}_{p_i}$ and $\bigcup_j \mathbf{r}_{p_j} = \mathbf{r}_{p_i}$
- (6) $\forall j \in [1, t]$ such that $\mathbf{r}_{p_j} \subset \mathbf{r}_{p_i} : \bigcup_j \mathbf{r}_{p_j} \neq \mathbf{r}_{p_i}$.

We define now the sets I_1, \dots, I_N and J_1, \dots, J_N with $|I_i| \leq |\mathcal{P}_i|$ and $|J_i| \leq |\mathcal{P}_i|$ for every $i = 1, \dots, N$ such that the values of the wires p_h can be perfectly simulated given the values $(a_i^{(1)})_{i \in I_1}, (b_i^{(1)})_{i \in J_1}, \dots, (a_i^{(N)})_{i \in I_N}, (b_i^{(N)})_{i \in J_N}$. The procedure to construct the sets is the following:

- For every wire in group (1), (2), (4) or (5), add the index of the shares of $\mathbf{a}^{(j)}$ in I_j .

We note that, since for every probe we add at most one input share, $|I_j| \leq |\mathcal{P}_i|$ for all j .

We now proceed with the simulation, consistently with the randomness involved.

- First of all, we pick at random every component of \mathbf{r}_{p_i} .
- Then we simulate every probe, by using the respective vector of random bits defined in the first phase and by using the share of the input $(a_i^{(j)})_{i \in I_j}$, which exist by construction.

This completes the proof. \square

Lemma 3.14. *The refreshing scheme in Algorithm 3.6 is $t - \text{SNI}$.*

Proof. We now prove the $t - \text{SNI}$ property of \mathcal{R}' .

Let $\Omega = (\mathcal{I}, \mathcal{O})$ be a set of t observations respectively on the internal and on the output wires, where $|\mathcal{I}| = t_1$ and in particular $t_1 + |\mathcal{O}| \leq t$.

We can classify the internal wires in the following groups:

- (1) a_i
- (2) $a_i + r_h, a_i r_h + r_k, a_i r_h + r_k + r_l$
- (3) $c_i = a_i r_h + r_k + r_l + r_m$

Suppose an adversary observes at most t wires w_1, \dots, w_t . We define two sets I with $|I| < t_1$ such that the values of the wires w_h can be perfectly simulated given the values $(a_i)_{i \in I}$. The procedure to construct the sets is the following:

- Initially I is empty and the w_i unassigned.
- For every wire in the group (1), (2), (3), (4) add i to I .

Note that by construction $|I| \leq t_1$.

We now proceed with the simulation of the wires w_h , using only the input shares $(a_i)_{i \in I}$.

- For each observation as in the group (1), $i \in I$ and then by definition of I the simulator has access to the value of a_i .
- For each observation as in the group (2), by construction $i \in I$ and if all the random bits have been probes, we can pick r_h (resp. r_h, r_k , or r_h, r_k, r_l) at random and simulate the value by using the share a_i . On the other hand, if there exists at least one random value which has not been observed, the probe can be sampled uniformly at random.

As for the output wires, we distinguish two cases. If some partial sum has already been observed, we remark that each output share involves the computation of t random bits and each of them appears a second time in a different output share. Therefore, since we have at most $t - 1$ additional probes to the current one, there exists at least one random value which has not been observed, the probe can be sample uniformly at random. \square

3.4. First-order security with constant amount of randomness

The first order ISW scheme is not particularly expensive in terms of randomness because it uses only one random bit. Unfortunately, when composed in more complicated circuits, the randomness involved increases with the size of the circuit, since we need fresh randomness for each gadget. Our idea is to avoid injecting new randomness in each multiplication and, instead, use alternatively the same random bits in all gadgets. We aim at providing a lower bound to the minimum number of bits needed in total to protect any circuit. Moreover, we show a matching upper bound, i.e., that it is possible to obtain a 1-probing secure private circuit, which uses only a constant amount of randomness. We emphasize that this means that the construction uses randomness that is *independent* of the circuit size, and, in particular, uses only 2 random bits in total per execution.

We will present a modified version of the usual gadgets for refreshing, multiplication, and the linear ones, which, in place of injecting new randomness, use a value taken from a set of two bits chosen at the beginning of each evaluation of the masked algorithm. We will design these schemes such that they will produce outputs depending on at most one random bit and such that every value in the circuit will be of a fixed form. The most challenging gadgets to treat are the multiplication and refreshing, as they are randomized and so responsible for the accumulation of randomness. On the other hand, even though the gadget for the addition does not use random bits, it will be subjected to some modifications as well, in order to avoid malicious situations that the reusing of the same random bits in the circuit can cause. As for the other linear gadgets, such as the powers $.^2$, $.^4$, and so on, they will be not affected by any change but will perform as usual share-wise computation.

We proceed by showing step by step the strategy to construct such circuits. First, we fix a set of bits $R = \{r_0, r_1\}$ where r_0 and r_1 are taken uniformly at random. The first randomized gadget of the circuit does not need to be substantially modified because there is no accumulation of randomness to be avoided yet. The only difference with the usual multiplication and refreshing gadgets is that, in place of the random component, we need to use one of the random bits in R , as shown in Algorithm 3.7 and Algorithm 3.8.

Secondly, we analyze the different configurations that an element can take when not

Algorithm 3.7 SecMult case (i)

Input: a_1, a_2 such that $a_1 + a_2 = a$, b_1, b_2 such that $b_1 + b_2 = b$

Output: r_k, c_i depending on a random number $r_k \in R$ such that $c_1 + c_2 = a \cdot b$

- 1: $r_k \xleftarrow{\$} R$;
 - 2: $c_1 \leftarrow a_1 b_1 + (a_1 b_2 + r_k)$;
 - 3: $c_2 \leftarrow a_2 b_1 + (a_2 b_2 - r_k)$;
-

Algorithm 3.8 Refreshing case (i)

Input: a_1, a_2 such that $a_1 + a_2 = a$
Output: r_k, c_i depending on the random number $r_k \in R$ such that $c_1 + c_2 = a$

- 1: $r_k \xleftarrow{\$} R$;
 - 2: $c_1 \leftarrow a_1 + r_k$;
 - 3: $c_2 \leftarrow a_2 - r_k$;
-

more than one randomized gadget has been executed, i.e., when only one random bit has been used in the circuit. The categories listed below represent the different forms that such an element takes if it is respectively the first input of the circuit, the output of the first refreshing scheme as in Algorithm 2.3, and the one of the first ISW multiplication scheme as in Algorithm 2.2 with inputs x and y :

- (1) $\mathbf{a} = (a_1, a_2)$;
- (2) $\mathbf{a} = (a_1 + r, a_2 - r)$, where r is a random bit in R ;
- (3) $\mathbf{a} = (x_1y_1 + x_1y_2 + r, x_2y_1 + x_2y_2 - r)$, where r is a random bit in R .

This categorization is important because, according to the different forms of the values that the second randomized gadget takes in input, the scheme accumulates randomness in different ways. Therefore, we need to modify the gadgets by taking into account the various possibilities for the inputs, i.e., distinguish if:

- (i) both the inputs are in category (1);
- (ii) the first input is as in category (1), i.e., $\mathbf{a} = (a_1, a_2)$, and the second one in category (2), i.e., $\mathbf{b} = (b_1 + r_1, b_2 - r_1)$;
- (iii) the first input is as in category (1), i.e., $\mathbf{a} = (a_1, a_2)$, and the second one in category (3), i.e., $\mathbf{b} = (c_1d_1 + c_1d_2 + r_1, c_2d_1 + c_2d_2 - r_1)$;
- (iv) the first input is in category (3), i.e., $\mathbf{a} = (c_1d_1 + c_1d_2 + r_0, c_2d_1 + c_2d_2 - r_0)$, and second one in category (2), i.e., $\mathbf{b} = (b_1 + r_1, b_2 - r_1)$;
- (v) both inputs are in category (2), i.e., $\mathbf{a} = (a_1 + r_1, a_2 - r_1)$ and $\mathbf{b} = (b_1 + r_0, b_2 - r_0)$;
- (vi) both inputs values are in category (3), i.e., $\mathbf{a} = (c_1d_1 + c_1d_2 + r_1, c_2d_1 + c_2d_2 - r_1)$ and $\mathbf{b} = (c'_1d'_1 + c'_1d'_2 + r_0, c'_2d'_1 + c'_2d'_2 - r_0)$.

where, for the moment, we suppose that the two inputs depend on two different random bits each. A more general scenario will be analyzed later. The goal of having the modified gadgets presented below is not only to reuse the same random bits, avoiding an accumulation at every execution, but also to produce outputs in the groups (1), (2) or (3), in order to keep such a configuration of the wires unchanged everywhere in the circuit. In this way, we guarantee that every wire depends only on one random bit and that we can use the same multiplication schemes in the entire circuit. According to this remark, we modify the ISW as depicted in Algorithm 3.9 and Algorithm 3.10.

It is easy to prove that the new multiplication algorithms are such that their outputs

Algorithm 3.9 SecMult case (ii) and (iii)

Input: a_1, a_2 such that $a_1 + a_2 = a$, b_1, b_2 depending on a random number $r_i \in R$ such that $b_1 + b_2 = b$, the set $R = \{r_0, r_1\}$, r_i

Output: r_{1-i} , c_i depending on the random number r_{1-i} such that $c_1 + c_2 = a \cdot b$

- 1: $c_1 \leftarrow a_1 b_1 + (a_1 b_2 + r_{1-i});$
 - 2: $c_2 \leftarrow a_2 b_1 + (a_2 b_2 - r_{1-i});$
-

Algorithm 3.10 SecMult case (iv), (v) and (vi)

Input: a_1, a_2 depending on the random number r_i such that $a_1 + a_2 = a$, b_1, b_2 depending on the random number r_{1-i} satisfying $b_1 + b_2 = b$, the set $R = \{r_0, r_1\}$

Output: r_{1-i} , c_i depending on the random number $r_{1-i} \in R$ satisfying $c_1 + c_2 = a \cdot b$

- 1: $\delta \leftarrow -r_{1-i};$
 - 2: $\delta \leftarrow \delta + r_i b_1;$
 - 3: $\delta \leftarrow \delta + r_i b_2;$
 - 4: $c_1 \leftarrow a_1 b_1 + (a_1 b_2 - \delta);$
 - 5: $c_2 \leftarrow a_2 b_1 + (a_2 b_2 + \delta);$
-

always belong to group (3).

Lemma 3.15. *Let a and b be two input values of Algorithm 3.9 or of Algorithm 3.10. Then, the output value $\mathbf{e} = \text{SecMult}(\mathbf{a}, \mathbf{b})$ is of the form (3).*

Proof. We show the proof of the lemma for the more complicated case of Algorithm 3.10. Case (iv): Let $\mathbf{a} = (c_1 d_1 + c_1 d_2 + r_0, c_2 d_1 + c_2 d_2 - r_0)$ and $\mathbf{b} = (b_1 + r_1, b_2 - r_1)$. Then, according to Algorithm 3.10, the output $\mathbf{e} = a \cdot b$ is such that:

$$\begin{aligned}
 e_1 &= (c_1 d_1 + c_1 d_2 + r_0)(b_1 + r_1) + ((c_1 d_1 + c_1 d_2 + r_0)(b_2 - r_1) - (r_1 + r_0 b_1 + r_0 b_2)) \\
 &= c_1 d_1 b_1 + c_1 d_2 b_1 + (c_1 d_1 b_2 + c_1 d_2 b_2 + r_1) \\
 e_2 &= (c_2 d_1 + c_2 d_2 - r_0)(b_1 + r_1) + ((c_2 d_1 + c_2 d_2 - r_0)(b_2 - r_1) + (r_1 + r_0 b_1 + r_0 b_2)) \\
 &= c_2 d_1 b_1 + c_2 d_2 b_1 + (c_2 d_1 b_2 + c_2 d_2 b_2 - r_1)
 \end{aligned}$$

Then e is in category (3).

Case (v): Let $\mathbf{a} = (a_1 + r_1, a_2 - r_1)$ and $\mathbf{b} = (b_1 + r_0, b_2 - r_0)$, then the product $\text{SecMult}(\mathbf{a}, \mathbf{b})$ is

$$\begin{aligned}
 e_1 &= (a_1 + r_1)(b_1 + r_0) + ((a_1 + r_1)(b_2 - r_0) - (r_0 + r_1 b_1 + r_1 b_2)) \\
 &= a_1 b_1 + a_1 r_0 + a_1 b_2 - a_1 r_0 + r_0 \\
 e_2 &= (a_2 - r_1)(b_1 + r_0) + ((a_2 - r_1)(b_2 - r_0) + (r_0 + r_1 b_1 + r_1 b_2)) \\
 &= a_2 b_1 + a_2 r_0 + a_2 b_2 - a_2 r_0 + r_0
 \end{aligned}$$

and then it is in category (3).

Algorithm 3.11 Modified refreshing \mathcal{R}'

Input: a_1, a_2 such that $a_1 + a_2 = a$ depending on a random bit r_i , the value r_i

Output: r_{1-i}, c_i depending on the random number r_{1-i} such that $c_1 + c_2 = a$

1: $c_1 \leftarrow (a_1 + r_{1-i}) - r_i;$

2: $c_2 \leftarrow (a_2 - r_{1-i}) + r_i;$

Case (vi): Let us suppose that $\mathbf{a} = (c_1d_1 + c_1d_2 + r_1, c_2d_1 + c_2d_2 - r_1)$ and that $\mathbf{b} = (c'_1d'_1 + c'_1d'_2 + r_0, c'_2d'_1 + c'_2d'_2 - r_0)$. Then the product $\mathbf{e} = \text{SecMult}(\mathbf{a}, \mathbf{b})$ is:

$$\begin{aligned} e_1 &= (c_1d_1 + c_1d_2 + r_1)(c'_1d'_1 + c'_1d'_2 + r_0) + ((c_1d_1 + c_1d_2 + r_1)(c'_2d'_1 + c'_2d'_2 - r_0) \\ &\quad - (r_0 + r_1c'_1d'_1 + r_1c'_1d'_2 + r_1c'_2d'_1 + r_1c'_2d'_2)) \\ &= c_1d_1c'_1d'_1 + c_1d_1c'_1d'_2 + c_1d_2c'_1d'_1 + c_1d_2c'_1d'_2 + c_1d_1c'_2d'_1 + c_1d_1c'_2d'_2 + c_1d_2c'_2d'_1 + c_1d_2c'_2d'_2 + r_0 \\ e_2 &= (c_2d_1 + c_2d_2 - r_1)(c'_1d'_1 + c'_1d'_2 + r_0) + ((c_2d_1 + c_2d_2 - r_1)(c'_2d'_1 + c'_2d'_2 - r_0) \\ &\quad + (r_0 + r_1c'_1d'_1 + r_1c'_1d'_2 + r_1c'_2d'_1 + r_1c'_2d'_2)) \\ &= c_2d_1c'_1d'_1 + c_2d_1c'_1d'_2 + c_2d_2c'_1d'_1 + c_2d_2c'_1d'_2 + c_2d_1c'_2d'_1 + c_2d_1c'_2d'_2 + c_2d_2c'_2d'_1 + c_2d_2c'_2d'_2 + r_0 \end{aligned}$$

and then it is in category (3). \square

As specified before, in the previous analysis we are supposed to have as input to the multiplication schemes values depending on different random bits. Since this is not always the case in practice, we need to introduce a modified refreshing scheme, which replaces the random bit on which the input depends, with the other random bit of the set R . The scheme is presented in Algorithm 3.11, and it has to be employed every time one of the input values of a multiplication scheme depends on the same randomness. Algorithm 3.11 is also useful before an addition gadget with inputs depending on the same random bit because it avoids that the randomness cancels out. The proof of correctness is quite straightforward, therefore we provide only an exemplary proof for a value in category (3).

Lemma 3.16. *Let a be an input value of the form (3) depending on a random bit $r_i \in R$ for Algorithm 3.11. Then the output value $\mathbf{e} := \mathcal{R}'(\mathbf{a})$ is of the form (3) and depends on the random bit r_{1-i} .*

Proof. Suppose without loss of generality that the input a depends on the random bit r_1 , so that $\mathbf{a} = (c_1d_1 + c_1d_2 + r_0, c_2d_1 + c_2d_1 - r_0)$. Then the output $\mathbf{e} := \mathcal{R}'(\mathbf{a})$ is:

$$\begin{aligned} e_1 &= (c_1d_1 + c_1d_2 + r_0 + r_1) - r_0 = c_1d_1 + c_1d_2 + r_1 \\ e_2 &= (c_2d_1 + c_2d_1 - r_0 - r_1) + r_0 = c_2d_1 + c_2d_1 - r_1 \end{aligned}$$

completing the proof. \square

Lastly, in Algorithm 3.12 we define a new scheme for addition, which allows to have outputs in one of the three categories (1), (2) or (3). Note that, thanks to the use of the

Algorithm 3.12 Modified addition Add'

Input: a_1, a_2 such that $a_1 + a_2 = a$ depending on a random bit r_i ,

b_1, b_2 such that $b_1 + b_2 = b$ depending on a random bit r_{1-i}

Output: $r_k \in R$, c_i depending on the random bit r_k such that $c_1 + c_2 = a + b$

1: $r_k \xleftarrow{\$} R$;

2: $c_1 \leftarrow a_1 + b_1 - r_k$;

3: $c_2 \leftarrow a_2 + b_2 + r_k$;

refreshing \mathcal{R}' , we can avoid having a dependence on the same random bit in the input of an addition gadget. The proof of correctness is again quite simple and therefore we provide only an exemplary proof for inputs as in case (iv).

Lemma 3.17. *Let a and b be two input values of Algorithm 3.12 as in case (iv). Then the output value $\mathbf{e} := \text{Add}'(\mathbf{a}, \mathbf{b})$ is in category (2).*

Proof. Suppose without loss of generality that the first input depends on r_0 and the second one on r_1 and that the value r_k picked at random in R during the execution of Algorithm 3.12 is r_1 . Let $\mathbf{a} = (c_1d_1 + c_1d_2 + r_0, c_2d_1 + c_2d_2 - r_0)$ and $\mathbf{b} = (b_1 + r_1, b_2 - r_1)$. Then according to Algorithm 3.12 the output $\mathbf{e} := \text{Add}'(\mathbf{a}, \mathbf{b})$ is:

$$e_1 = c_1d_1 + c_1d_2 + r_0 + b_1 + r_1 - r_0 = c_1d_1 + c_1d_2 + b_1 + r_1$$

$$e_2 = c_2d_1 + c_2d_2 - r_0 + b_2 - r_1 + r_0 = c_1d_1 + c_1d_2 + b_1 - r_1$$

completing the proof. □

In conclusion, we notice that by using the schemes above and composing them according to the instructions just given, we obtain a circuit where each wire carries a value of a fixed form (i.e., in one of the categories (1), (2) or (3)). Consequently, we can always use one of the multiplication schemes given in the Algorithms 3.7, 3.9, and 3.10 without accumulating randomness and without the risk of canceling the random bits out. Furthermore, it is easy to see that all the schemes just presented are secure against a 1-probing attack.

Counterexample for the 1-bit randomness case

In the following we show that it is impossible in general to have a 1st-order probing secure circuit, which uses only 1 bit of randomness in total. In particular, we present a counterexample which breaks the security of a circuit using only one random bit.

Let us restrict our attention to the \mathbb{F}_2 field and consider \mathbf{c} and \mathbf{c}' two outputs of two multiplication schemes between the values \mathbf{a}, \mathbf{b} and \mathbf{a}', \mathbf{b}' respectively, and let r be the

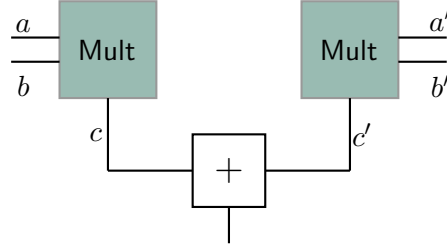


Figure 3.6.: Example of composition where the randomness can completely cancel out

only random bit which is used in the entire circuit. Then \mathbf{c} and \mathbf{c}' are of the form

$$\begin{cases} c_1 = a_1 b_1 \oplus a_1 b_2 \oplus r \\ c_2 = a_2 b_1 \oplus a_2 b_2 \oplus r \end{cases} \quad \text{and} \quad \begin{cases} c'_1 = a'_1 b'_1 \oplus a'_1 b'_2 \oplus r \\ c'_2 = a'_2 b'_1 \oplus a'_2 b'_2 \oplus r \end{cases}.$$

Suppose now that these two values are inputs of an additive gadget, as in Figure 3.6. Such a gadget could either use no randomness at all and just add the components to each other, or involve in the computation the bit r maintaining the correctness. In the first case we obtain

$$\begin{aligned} c_1 \oplus c'_1 &= a_1 b_1 \oplus a_1 b_2 \oplus a'_1 b'_1 \oplus a'_1 b'_2 = a_1 b \oplus a'_1 b' \\ c_2 \oplus c'_2 &= a_2 b_1 \oplus a_2 b_2 \oplus a'_2 b'_1 \oplus a'_2 b'_2 = a_2 b \oplus a'_2 b' \end{aligned}$$

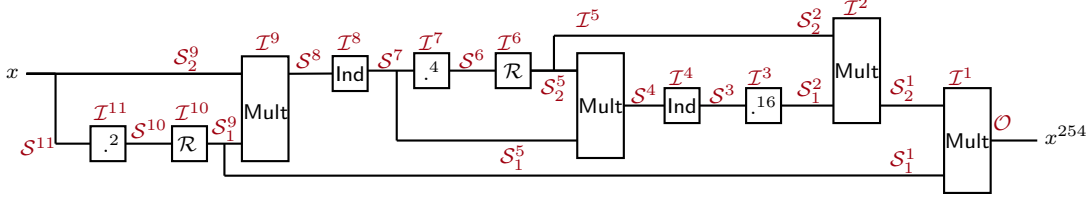
and then the randomness r will be completely canceled out, revealing the secret. In the second case, if we inject in the computation another r , then, in whatever point of the computation we put it, it will cancel out again one of the two r revealing one of the secrets during the computation of the output. For example, we can have

$$\begin{aligned} c_1 \oplus c'_1 &= r \oplus a_1 b_1 \oplus a_1 b_2 \oplus r \oplus a'_1 b'_1 \oplus a'_1 b'_2 \oplus r = a_1 b \oplus a'_1 b'_1 \oplus a'_1 b'_2 \oplus r \\ c_2 \oplus c'_2 &= r \oplus a_2 b_1 \oplus a_2 b_2 \oplus r \oplus a'_2 b'_1 \oplus a'_2 b'_2 \oplus r = a_2 b \oplus a'_2 b'_1 \oplus a'_2 b'_2 \oplus r \end{aligned}$$

In view of this counterexample, we can conclude that the minimum number of random bits needed in order to have a 1st-order private circuit is 2.

3.5. Case study: AES

To evaluate the impact of our methodology on the performance of protected implementations, we present the implementation of the AES-128 without and with common randomness performed in [FPS17]. In particular, we consider the inversion of each Sbox call as a block of gadgets $\mathcal{G}_{i=1,\dots,200}$ using the same random components and the last multiplication of each of these inversions is using fresh randomness. We use Rivain and


 Figure 3.7.: Non-linear part of the AES S-box with the insertion of the `lnd` gadget

Prouff’s algorithms chain from [Cor+13; RP10], depicted in Figure 2.6. For the implementation without common randomness, we use the multiplication algorithm from [RP10] and the refreshing from [DDF14] (cf. Algorithm 2.3). To enable the use of common randomness, we replace the multiplication with our t – SCR multiplication schemes, and we place the `lnd` gadgets where needed. In particular, since the inputs of the first call of the inversion are independent, we do not need to add `lnd` before the first multiplication scheme with common randomness, while this is needed for the next multiplications in the circuit. On the other hand, the last multiplication scheme of each inversion needs to be completely refreshed, because the insertion of the gadgets `lnd` for guaranteeing the independence of the inputs would make the circuit insecure and not t – SNI. For this reason, the outputs of each inversion are independent again and the S-box is composable. Therefore the same procedure can be used for each round. The algorithm is depicted in Figure 3.7 and Lemma 3.18 shows its security.

Lemma 3.18. *Gadget \cdot^{254} , depicted in Figure 3.7, is t – SNI.*

Proof. Let $\Omega = (\bigcup_{1 \leq i \leq 11}, \mathcal{O})$ be a set of t probes such that $\sum_{1 \leq i \leq 11} |\mathcal{I}^i| \leq t_1$ $|\mathcal{O}| + t_1 \leq t$. In the following, we show the existence of a simulator which simulates the probes by using at most t_1 internal values. The simulation is processed from right to left, according to the enumeration in Figure 3.7.

Gadget 1: Since `Mult` is t – SNI and $|\mathcal{I}^1 \cup \mathcal{O}| \leq t$, then there exist two sets of indexes \mathcal{S}_1^1 and \mathcal{S}_2^1 such that $|\mathcal{S}_1^1| \leq |\mathcal{I}^1|$, $|\mathcal{S}_2^1| \leq |\mathcal{I}^1|$ and the gadget can be perfectly simulated from its input shares corresponding to the indexes in \mathcal{S}_1^1 and \mathcal{S}_2^1 .

Gadget 2: Since `Mult` is t – SNI and $|\mathcal{I}^2 \cup \mathcal{S}_2^1| \leq t$, then there exist two sets of indexes \mathcal{S}_1^2 and \mathcal{S}_2^2 such that $|\mathcal{S}_1^2| \leq |\mathcal{I}^2|$, $|\mathcal{S}_2^2| \leq |\mathcal{I}^2|$ and the gadget can be perfectly simulated from its input shares corresponding to the indexes in \mathcal{S}_1^2 and \mathcal{S}_2^2 .

Gadget 3: Since \cdot^{16} is a linear gadget, then there exists a set of indexes \mathcal{S}^3 such that $|\mathcal{S}^3| \leq |\mathcal{I}^3| + |\mathcal{S}_1^2| \leq |\mathcal{I}^3| + |\mathcal{I}^2|$ and the gadget can be perfectly simulated from its input shares corresponding to the indexes in \mathcal{S}^3 .

Gadget 4: Since `lnd` is t – NI then there exists a set of indexes \mathcal{S}^4 such that $|\mathcal{S}^4| \leq |\mathcal{I}^4| + |\mathcal{S}^3| \leq |\mathcal{I}^4| + |\mathcal{I}^3| + |\mathcal{I}^2|$ and the gadget can be perfectly simulated from its input shares corresponding to the indexes in \mathcal{S}^4 .

<i>W/o Common Randomness</i>				<i>W/ Common Randomness</i>		
t	n	Mult.	Refresh	n	Mult.	Refresh
1	2	1	1	2	1	1
2	3	3	3	3	3	3
3	4	6	6	4	6	6
4	5	10	10	7	18	14
5	6	15	15	7	21	18

Table 3.2.: Number of random elements required for the multiplication and refresh algorithms with and without common randomness from $t = 1$ to $t = 5$

Gadget 5: Since Mult is $t - \text{SNI}$ and $|\mathcal{I}^5 \cup \mathcal{S}^4| \leq t$, then there exist two sets of indexes \mathcal{S}_1^5 and \mathcal{S}_2^5 such that $|\mathcal{S}_1^5| \leq |\mathcal{I}^5|$, $|\mathcal{S}_2^5| \leq |\mathcal{I}^5|$ and the gadget can be perfectly simulated from its input shares corresponding to the indexes in \mathcal{S}_1^5 and \mathcal{S}_2^5 .

Gadget 6: Since \mathcal{R} is $t - \text{SNI}$ and $|\mathcal{I}^6 \cup \mathcal{S}_2^5| \leq t$, then there exist a set of indexes \mathcal{S}^6 such that $|\mathcal{S}^6| \leq |\mathcal{I}^6|$ and the gadget can be perfectly simulated from its input shares corresponding to the indexes in \mathcal{S}^6 .

Gadget 7: Since \cdot^4 is a linear gadget then there exists a set of indexes \mathcal{S}^7 such that $|\mathcal{S}^7| \leq |\mathcal{I}^7| + |\mathcal{S}^6| \leq |\mathcal{I}^7| + |\mathcal{I}^6|$ and the gadget can be perfectly simulated from its input shares corresponding to the indexes in \mathcal{S}^7 .

Gadget 8: Since Ind is $t - \text{NI}$ then there exists a set of indexes \mathcal{S}^8 such that $|\mathcal{S}^8| \leq |\mathcal{I}^8| + |\mathcal{S}^7| \leq |\mathcal{I}^8| + |\mathcal{I}^7| + |\mathcal{I}^6|$ and the gadget can be perfectly simulated from its input shares corresponding to the indexes in \mathcal{S}^8 .

Gadget 9: Since Mult is $t - \text{SNI}$ and $|\mathcal{I}^9 \cup \mathcal{S}^8| \leq t$, then there exist two sets of indexes \mathcal{S}_1^9 and \mathcal{S}_2^9 such that $|\mathcal{S}_1^9| \leq |\mathcal{I}^9|$, $|\mathcal{S}_2^9| \leq |\mathcal{I}^9|$ and the gadget can be perfectly simulated from its input shares corresponding to the indexes in \mathcal{S}_1^9 and \mathcal{S}_2^9 .

Gadget 10: Since \mathcal{R} is $t - \text{SNI}$ and $|\mathcal{I}^{10} \cup \mathcal{S}_1^9| \leq t$, then there exist a set of indexes \mathcal{S}^{10} such that $|\mathcal{S}^{10}| \leq |\mathcal{I}^{10}|$ and the gadget can be perfectly simulated from its input shares corresponding to the indexes in \mathcal{S}^{10} .

Gadget 11: Since \cdot^2 is linear then there exists a set of indexes \mathcal{S}^{11} such that $|\mathcal{S}^{11}| \leq |\mathcal{I}^{11}| + |\mathcal{S}^{10}| \leq |\mathcal{I}^{11}| + |\mathcal{I}^{10}|$ and the gadget can be perfectly simulated from its input shares corresponding to the indexes in \mathcal{S}^{11} .

Each of the previous steps show the existence of a simulator each of the respective gadgets and by composing them we obtain a global simulator of the entire circuit which uses $|\mathcal{S}^{11} \cup \mathcal{S}_2^9|$ shares of the input. Since $|\mathcal{S}^{11} \cup \mathcal{S}_2^9| \leq |\mathcal{I}^{11}| + |\mathcal{I}^{10}| + |\mathcal{I}^9| \leq t_1$ we conclude that the gadget \cdot^{254} is $t - \text{SNI}$. \square

Without Common Randomness					With Common Randomness			
t	n	TRNG	Cycle Count		n	TRNG	Cycle Count	
		Calls	TRNG ₃₂	TRNG ₈		Calls	TRNG ₃₂	TRNG ₈
1	2	1,200	112,919	187,519	2	2	73,441	73,560
					2	605	86,137	99,334
					3	1,415	200,423	230,334
2	3	3,600	308,600	548,477	3	1,415	200,423	230,334
3	4	7,200	496,698	1,089,092	4	2,430	292,579	412,523

Table 3.3.: Cycle counts of our AES implementations on an ARM Cortex-M4F with TRNG₃₂. In addition, we provide the required number of calls to the TRNG for each t

Table 3.2 summarizes the randomness requirements of both types of refresh and multiplication algorithms presented in the chapter. We point out that the implementation of the S-box is performed only up to order $t = 3$, since the schemes for higher orders are only weakly t -SCR, and therefore cannot be composed while sharing randomness.

In [FPS17], both types of protected AES were implemented on an ARM Cortex-M4F running at 168 MHz using C. The random components were generated using the TRNG of the evaluation board (STM32F4 DISCOVERY) which generates 32 bits of randomness every 40 clock cycles running in parallel at 48 MHz. To assess the influence of the TRNG performance on the result, the authors considered two modes of operation for the randomness generation: a fast TRNG₃₂ and a slower TRNG₈. The problem of randomness generation affects a majority of implementations independently of the degree of optimization and can pose a bottleneck, especially if no dedicated TRNG is available. Therefore, the performance results provided can be transferred to other types of implementations and platforms. As shown in Table 3.3, the implementations with common randomness requires fewer calls to the TRNG.

For the special case of $t = 1$, we presented a solution (cf. Section 3.4) with constant randomness independent of the circuit size. Following the aforementioned procedure, we realized a 1-probing secure AES implementation with only two TRNG calls. Overall, the implementation using the constant randomness scheme requires less cycles than the one with common randomness. In general, however, this advantage strongly depends on the performance of the TRNG as the implementation with constant randomness requires additional operations to ensure security.

3.6. Conclusions

We proposed the new security notions of t -SCR and weak- t -SCR, for gadgets that have part of the internal randomness in common. The new definition provides the requisites to reuse randomness securely among multiple gadgets (or sets of gadgets). We showed a method to design multiplication schemes secure under these notions. Since the number of n for our t -SCR multiplication grows in $\mathcal{O}(t^2)$ and our multiplication algorithm (resp. \mathcal{R}) requires $\mathcal{O}(n^2)$ (resp. $\mathcal{O}(nt)$) random elements, the practicability of our proposed methodology becomes limited for increasing t . An additional limitation is given by the fact that it is possible to build t -SCR multiplication schemes only up to order 3. Nevertheless, our case study showed that for small $t \leq 3$ our approach results in significant performance improvement for the masked implementations. The improvement factor could potentially be even larger, if we replace our efficient TRNG with a common PRNG.

Another interesting aspect for future work is designing an automatic application of our methodology to an arbitrary circuit. While we showed how to use our methodology on a circuit with a layered structure which contains easily-exploitable regularities, further research might be able to derive an algorithm which finds the optimal grouping for any given design. This would help to create a compiler which automatically applies masking to an unprotected architecture in the most efficient way removing the requirement for a security-literate implementer and reducing the chance for human error.

4. Randomness optimization in Inner Product masking

In this chapter, we move our attention to another type of masking scheme, that exhibits a higher security level, at the cost of reduced performances: the Inner Product (IP) masking.

Motivations

As already mentioned before in this work, the most employed masking scheme in protected implementations is the Boolean masking, thanks to its simplicity and comparably low complexity overheads. Nevertheless, several works have proposed masking schemes using alternative encoding functions. Such different masking schemes can be compared to each other by using the framework proposed in [SMY09]. According to it, the mutual information between a secret variable and its corresponding leakages can serve as a figure of merit for side-channel security, since it is proportional to the success rate of a (worst-case) Bayesian adversary exploiting these leakages (see [DDF14]).

An example of a masking scheme different from the Boolean one is the Inner Product masking scheme, introduced by Balasch *et al.* [Bal+12], and based on the inner product construction of Dziembowski and Faust [DF12]. According to its encoding function, a sensitive variable S is masked into $2 \cdot n$ shares given by two random vectors (\mathbf{L}, \mathbf{S}) of n elements each, such that $S = \langle \mathbf{L}, \mathbf{S} \rangle$. Interestingly, applying the evaluations from [SMY09], it emerges that, for low noise levels, the Inner Product masking achieves lower information leakage compared to other types of masking at the same security order t . Additionally, the multiplication scheme proposed in [Bal+12] achieves the complexity of $O(n^2)$, hence similar to the one for Boolean masking. Despite the advantages just mentioned, the first version of the Inner Product masking suffers from the efficiency point of view since the addition and refreshing algorithms require larger constant terms than their analogs in the Boolean masking schemes.

After the first version of the Inner Product masking presented in [DF12], Balasch *et al.* in [BFG15] improved the encoding function, by letting L be a public value with first element $L_1 = 1$. This change reduces the number of secret shares from $2n$ to n and results as well in excellent efficiency improvements for all masked operations. Specifically, the complexity of the addition and refreshing schemes is comparable to those of Boolean and Polynomial masking schemes. However, the authors did not provide security proofs

according to the properties of $t - \text{NI}$ and $t - \text{SNI}$ [Bar+16], which allows to compose the algorithms securely.

Our work has the goal of increasing the practicality of the Inner Product masking scheme and providing novel insights into it. Our main contribution is to consolidate the existing works of Balasch *et al.* [Bal+12; BFG15], by mainly improving the previous algorithms in terms of efficiency and security guarantees. Moreover, in [Bal+17], we provide a further investigation on the information theoretic evaluation of the Inner Product masking as well as implementation results and experimental evaluations, which we briefly mention at the end of this chapter.

Contribution

We describe our contribution, focusing on the main concepts.

New efficient algorithms satisfying the $t - \text{SNI}$ security property. We propose simplified algorithms for the multiplication operation protected with Inner Product masking. In contrast to the schemes from [Bal+12; BFG15], our algorithms resemble the schemes originally proposed by Ishai *et al.* [ISW03], but handle the Inner Product encoding function. Thanks to this similarity, they are more efficient and easier to implement than the schemes in [BFG15]. Additionally, we prove that our new algorithms satisfy the property of t -Strong Non-Interference ($t - \text{SNI}$) introduced by Barthe *et al.* [Bar+16], and hence can be easily composed in a secure circuit.

To give an intuition about how our algorithms work and why they constitute an improvement over previous schemes, in the following, we shortly present the multiplication scheme at 2^{nd} order of security. In particular, we start by recalling the multiplication scheme introduced in [BFG15], in order to underline the main differences with ours.

Every masked implementation initially starts by a set-up phase, where we define a random vector \mathbf{L} , whose first component is set to 1 and all the others are non zero random elements in a field \mathcal{K} of characteristic 2. The two inputs $\mathbf{A}, \mathbf{B} \in \mathcal{K}$ of the multiplication scheme are masked as $\mathbf{A} = (A_1, A_2, A_3)$ and $\mathbf{B} = (B_1, B_2, B_3)$ such that $A = \langle \mathbf{L}, \mathbf{A} \rangle$ and $B = \langle \mathbf{L}, \mathbf{B} \rangle$. The encoded output of the multiplication between A and B is $\mathbf{C} = (C_1, C_2, C_3)$ and it is defined as in Figure 4.1, where the elements $U_{i,j}$, with $i, j = 1, 2, 3$, are sampled uniformly at random from the field \mathcal{K} . The algorithm requires the use of $n^2 = 9$ random elements, additionally to the ones used in the set-up phase.

In order to improve the efficiency of such a multiplication scheme, we completely change the algorithm in [BFG15], and we rather follow an approach analog to the one of Ishai *et al.* [ISW03]. Similarly, first we construct a matrix containing all the products of the input shares $A_i \cdot B_j$, and a symmetric matrix of random elements. Then, we sum component-wise the matrices, and we assign the sum of the elements on each row to the output shares. When applying this technique to the Inner Product masking, the

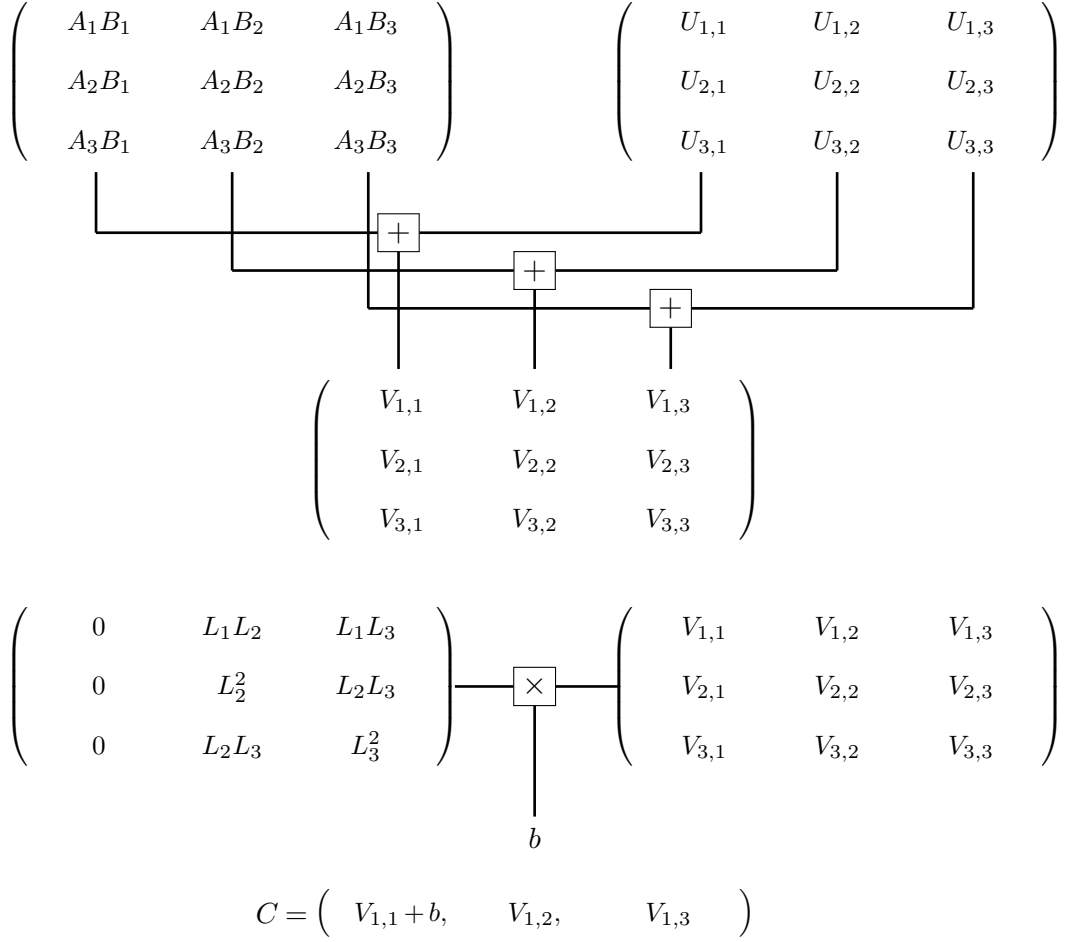
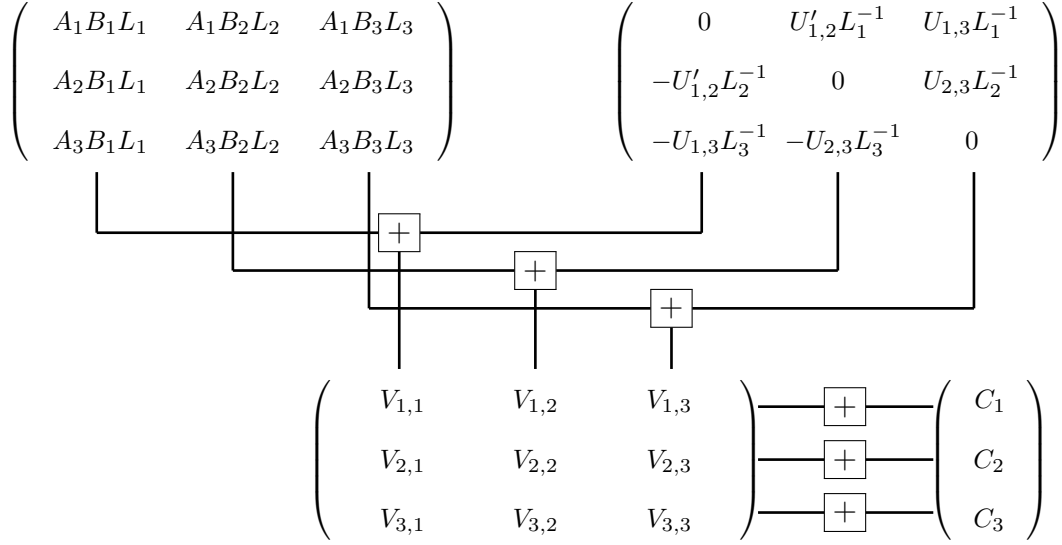


Figure 4.1.: Multiplication scheme by [BFG15] for $t = 2$ with inputs A, B and output C

main challenge with respect to the Boolean masking is to take into account the vector \mathbf{L} , such that the correctness holds, i.e., that $\mathbf{C} = \langle \mathbf{L}, \mathbf{A} \rangle \cdot \langle \mathbf{L}, \mathbf{B} \rangle$. To this end, we multiply each A_iB_j with L_j , and each random element $U'_{i,j}$ with L_i and $-U'_{i,j}$ with L_j . Our new multiplication scheme, which we name $\text{IPMult}_{\mathbf{L}}^{(1)}$, is depicted in Figure 4.2, where the elements $U'_{i,j}$, with $i = 1, 2$ and $j = 2, 3$, are sampled at random from the field \mathcal{K} . The correctness comes from the following calculation:

$$\begin{aligned}
\langle \mathbf{L}, \mathbf{C} \rangle &= L_1C_1 + L_2C_2 + L_3C_3 = \\
&= L_1(V_{1,1} + V_{1,2} + V_{1,3}) + L_2(V_{2,1} + V_{2,2} + V_{2,3}) + L_3(V_{3,1} + V_{3,2} + V_{3,3}) = \\
&= L_1(A_1B_1 + A_1B_2 + A_1B_3 + U'_{1,2}L_1^{-1} + U'_{1,3}L_1^{-1}) + L_2(A_2B_1L_1 - U'_{1,2}L_2^{-1} + A_2B_2L_2 \\
&\quad + U'_{2,3}L_2^{-1} + A_2B_3L_3) + L_3(A_3B_1L_1 - U'_{1,3}L_3^{-1} + A_3B_2L_3 - U'_{2,3}L_3^{-1} + A_3B_3L_3) \\
&= L_1(A_1B_1 + A_1B_2L_2 + A_1B_3L_3) + L_2(A_2B_1 + A_2B_2L_2 + A_2B_3L_3)
\end{aligned}$$


 Figure 4.2.: New $\text{IPMult}^{(1)}$ for $t = 2$ with inputs A, B and output C

$$\begin{aligned}
 &+ L_3(A_3 B_1 + A_3 B_2 L_2 + A_3 B_3 L_3) + U'_{1,2} + U'_{1,3} - U'_{1,2} + U'_{2,3} - U'_{1,3} - U'_{2,3} \\
 &= (L_1 A_1 + L_2 A_2 + L_3 A_3)(L_1 B_1 + L_2 B_2 + L_3 B_3) = \langle \mathbf{L}, \mathbf{A} \rangle \langle \mathbf{L}, \mathbf{B} \rangle
 \end{aligned}$$

It is easy to see that in our $\text{IPMult}_L^{(1)}$ algorithm there is a drop in the use of randomness compared to the algorithm in [BFG15]. In particular, our scheme employs only $\frac{n(n-1)}{2} = 3$ random elements, in contrast to the $n^2 = 9$ needed in [BFG15].

To this first contribution, we add a second multiplication algorithm, $\text{IPMult}^{(2)}$, that shows better efficiency when composed with linear operations. Concretely, when we want to compose a multiplication with a linear function g of one of its inputs, then, in order to guarantee independence between the inputs of the multiplication, we can use $\text{IPMult}_L^{(1)}$ and require an additional refreshing operation at the output of g , or we use our new algorithm $\text{IPMult}_L^{(2)}$ that eliminates the need for the additional refreshing. We will see that this strategy can save at least $\mathcal{O}(n^2)$ random elements.

Again, we briefly show the algorithm at 2^{nd} security order. The input \mathbf{B} is represented by $g(\mathbf{A})$, i.e., the evaluation of a linear function g on the first input A . The core of our idea is to internally refresh the input \mathbf{A} by adding to each share A_i a random element u_j such that the inputs become independent. The rest of the computation is the same as $\text{IPMult}_L^{(1)}$, with the only difference that, before generating the matrix of the $V_{i,j}$, we need to add a matrix of elements $g(A_j)u_j L_j$ and 0 on the diagonal, which guarantees the correctness. The complete algorithm is represented in Figure 4.3. We point out that our approach requires only n extra random elements. On the other hand, if we use the $\text{IPMult}_L^{(1)}$ we need to additionally refresh one of the inputs via a t -SNI refreshing scheme, which means using $n(n-1)$ extra random elements.

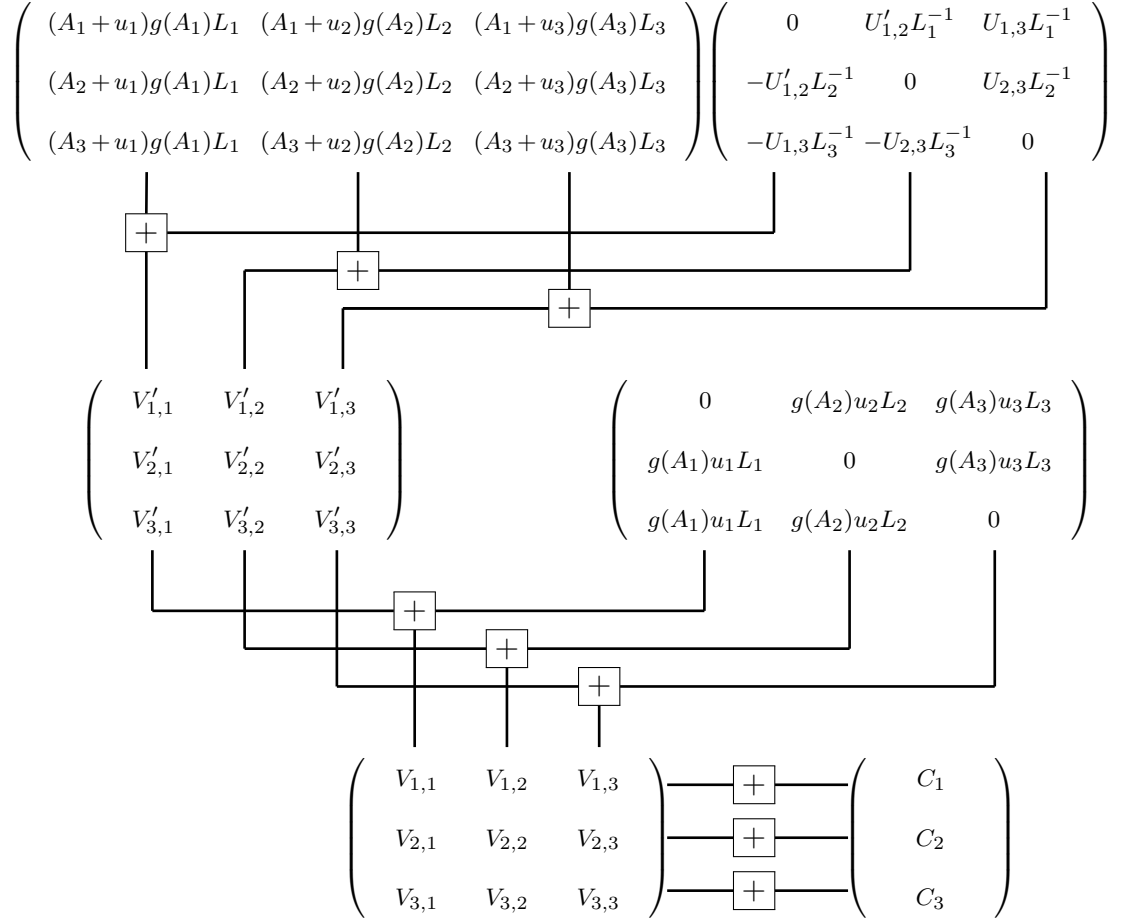


Figure 4.3.: New $\text{IPMult}^{(2)}$ for $t = 2$ with inputs A, B and output C

Complementary analysis from [Bal+17]. We discuss the comprehensive analysis conducted in [Bal+17] on our new construction and Inner Product masking in general. The implementation results show a reduction in the execution times by approximately 60% (for 2 shares) and 55% (for 3 shares) compared to earlier works [BFG15]. In information theoretic evaluation, that examines the mutual information between a secret variable and its corresponding leakages, shows that, in some cases, the increased algebraic complexity of Inner Product masking gives a good impact on the security order in the bounded moment model. Finally, the leakage detection techniques adopted in the experimental evaluations reveal that Inner Product masking present notably less evidence of leakage than Boolean masking.

Related work

Several other works in the literature study countermeasures against SCA based on other types of masking schemes, than the popular Boolean masking. Von Willich [von01] proposed the Affine masking, which encodes a variable X as $S = R_1 \cdot X + R_2$, where R_1 is a uniformly distributed non-zero multiplicative mask and R_2 a uniformly distributed additive mask. The advantage of their masking scheme is based on the observation that a higher algebraic complexity of the shares results in reduced information leakage. Fumaroli *et al.* [Fum+10] present an implementation of the AES using Affine masking, which shows performance results similar to the ones of Boolean masking. However, a drawback of Affine masking is that it has not been generalized to high-order.

Another proposed different type of masking is the Polynomial masking. Independently introduced by Prouff and Roche [PR11] and Goubin and Martinelli [GM11], it is based on Shamir's secret-sharing [Sha79] and secure multi-party computation techniques [BGW88]. Every sensitive variable X is associated to a polynomial of degree the security order t of the form $p_X = X + \sum_{i=1}^t a_i \cdot X^i$, where the a_i are random secret coefficients. In order to encode X we select n distinct non-zero elements α_i and evaluate $X_i = p_X(\alpha_i)$ for $i = 1, \dots, n$. The mask of X consists of $(\alpha_1, X_1), \dots, (\alpha_n, X_n)$. The decoding process consists in the computation of $S = \sum_{i=1}^n X_i \cdot \beta_i$, where the coefficients β_i are computed from the public α_i . The multiplication algorithms presented in the two papers, despite different from each others, have the same complexity in the number of shares, which is $O(n^3)$. Later in [Cor+13], Coron *et al.* improved this complexity to $O(n^2)$ by using the discrete Fourier transform for fast polynomial evaluation, achieving the same complexity of Boolean masking.

Notation

In the following we denote by \mathcal{K} a field of characteristic 2. We denote with upper-case letters the elements of the field \mathcal{K} and with bold notation the elements of the \mathcal{K} -vector space. The field multiplication is represented by the dot \cdot while the standard inner product over \mathcal{K} is denoted as $\langle \mathbf{X}, \mathbf{Y} \rangle = \sum_i X_i \cdot Y_i$, where X_i and Y_i are the components of the vectors \mathbf{X} and \mathbf{Y} .

4.1. A new multiplication scheme for IP masking

We present in this section the algorithms for the protection of a circuit using the Inner Product masking, and, in particular, we show the first of our new multiplication schemes.

Each masked implementation is associated to a vector \mathbf{L} , which is a fixed but random non-zero parameter with first component $L_1 = 1$. We recall that the Inner Product encoding of a variable $S \in \mathcal{K}$ consists of a vector $\mathbf{S} \in \mathcal{K}^n$ such that $S = \langle \mathbf{L}, \mathbf{S} \rangle$. Concretely,

the algorithms for initialization and masking are depicted in the **IPSetup** and **Encode** procedures. The subroutine $\text{rand}(\mathcal{K})$ samples an element uniformly at random from the field \mathcal{K} . The algorithm for addition is kept the same as in [BFG15] and it is depicted in Algorithm 4.3.

As for the refreshing scheme, we use the one from [BFG15], **IPRefresh_L** represented in Algorithm 4.4, which was proven to be t -probing secure, and we design a new t -SNI refreshing **SecIPRefresh_L**. This essentially consists of the execution of **IPRefresh_L** n times. In [BFG15] the authors already remarked that such a scheme ensures security even if composed with other gadgets, but no formal proof was provided. In the following we formally analyze the security of the algorithm, by giving the proof of t -SNI.

As for the multiplication, we propose a new scheme **IPMult⁽¹⁾** in Algorithm 4.6. The symbol δ_{ij} corresponds to the element 0 when $i = j$ and 1 otherwise. The scheme achieves t -SNI security. Our starting point for the construction of the algorithm is the multiplication scheme from [ISW03]. We reuse the idea of summing the matrix of the products of the inputs with a symmetric matrix of random elements, in order to compute the shares of the output in a secure way. In particular we design these two matrices (**T** and **U'** in the algorithm) to be consistent with our different masking scheme.

The correctness of the scheme is proved in the following lemma.

Lemma 4.1 (Correctness). *For any $\mathbf{L}, \mathbf{A}, \mathbf{B} \in \mathcal{K}^n$ and $\mathbf{C} = \text{IPMult}_{\mathbf{L}}^{(1)}(\mathbf{A}, \mathbf{B})$, we have*

$$\langle \mathbf{L}, \mathbf{C} \rangle = \langle \mathbf{L}, \mathbf{A} \rangle \cdot \langle \mathbf{L}, \mathbf{B} \rangle.$$

Proof. For all $i \neq j$ it holds:

$$\begin{aligned} \langle \mathbf{L}, \mathbf{C} \rangle &= \sum_i L_i \cdot C_i = \sum_i L_i \sum_j V_{i,j} = \sum_i L_i \sum_j (T_{ij} + U_{ij}) \\ &= \sum_i L_i \sum_j (A_i B_j L_j + U'_{ij} L_i^{-1}) = \sum_i L_i \sum_j A_i B_j L_j + \sum_{ij} U'_{ij} \\ &= \sum_i L_i A_i \sum_j B_j L_j = \langle \mathbf{L}, \mathbf{A} \rangle \langle \mathbf{L}, \mathbf{B} \rangle. \end{aligned}$$

□

Algorithm 4.1 Setup the masking scheme: $\mathbf{L} \leftarrow \text{IPSetup}_n(\mathcal{K})$

Input: field description \mathcal{K}

Output: random vector \mathbf{L}

- 1: $L_1 = 1$;
 - 2: **for** $i = 2$ to n **do**
 - 3: $L_i \leftarrow \text{rand}(\mathcal{K} \setminus \{0\})$;
-

Algorithm 4.2 Masking a variable: $\mathbf{S} \leftarrow \text{Encode}_{\mathbf{L}}(S)$

Input: variable $S \in \mathcal{K}$

Output: vector \mathbf{S} such that $S = \langle \mathbf{L}, \mathbf{S} \rangle$

- 1: **for** $i = 2$ to n **do**
 - 2: $S_i \leftarrow \text{rand}(\mathcal{K});$
 - 3: $S_1 = S + \sum_{i=2}^n L_i \cdot S_i;$
-

Algorithm 4.3 Add masked values: $\mathbf{C} \leftarrow \text{Add}(\mathbf{A}, \mathbf{B})$

Input: vector \mathbf{A}, \mathbf{B}

Output: vector \mathbf{C} such that $\langle \mathbf{L}, \mathbf{C} \rangle = \langle \mathbf{L}, \mathbf{A} \rangle + \langle \mathbf{L}, \mathbf{B} \rangle$

- 1: **for** $i = 1$ to n **do**
 - 2: $C_i \leftarrow A_i + B_i;$
-

Security analysis

We analyze the security of our new multiplication scheme in the t -probing model, and in particular we prove the stronger property of t -Strong Non-Interference (t -SNI) defined by Barthe *et al.* in [Bar+15a] and recalled in Definition 2.5.

The following lemma proves the security of the refreshing scheme $\text{SecIPRefresh}_{\mathbf{L}}$.

Lemma 4.2. *The algorithm $\text{SecIPRefresh}_{\mathbf{L}}$ is t -SNI with $t = n - 1$.*

Proof. Let $\Omega = (\mathcal{I}, \mathcal{O})$ be a set of t observations respectively on the internal and on the output wires, where $|\mathcal{I}| = t_1$ and in particular $t_1 + |\mathcal{O}| \leq t$. We point out the existence of a perfect simulator of the adversary's probes, which makes use of at most t_1 shares of the secret \mathbf{X} .

The internal wires w_h are classified as follows:

- (1) X_i
- (2) $R_{i,j}$, which is the component i of the vector \mathbf{R} in the j^{th} $\text{IPRefresh}_{\mathbf{L}}$
- (3) $Y_{i,j} = X_i + \sum_{k=1}^j R_{i,k}$, which is the component i of \mathbf{Y} in the j^{th} $\text{IPRefresh}_{\mathbf{L}}$

We define a set of indices I such that $|I| \leq t_1$ as follows: for every observation as in group (1), (2) or (3) add i to I .

Now we construct a simulator that makes use only of $(X_i)_{i \in I}$.

- For each observation as in group (1), $i \in I$ and then by definition of I the simulator has access to the value of X_i .
- For each observation as in group (2), $R_{i,j}$ can be sampled uniformly at random. Indeed, this is what happens in the real execution of the algorithm for the shares $R_{i,j}$ with $i = 2, \dots, n$. Otherwise, since we have at most $n - 1$ probes, the adversary's view of $R_{i,j}$ is also uniformly random.

Algorithm 4.4 Refresh vector: $\mathbf{X}' \leftarrow \text{IPRefresh}_{\mathbf{L}}(\mathbf{X})$ [BFG15]

Input: vector \mathbf{X}
Output: vector \mathbf{X}' such that $\langle \mathbf{L}, \mathbf{X} \rangle = \langle \mathbf{L}, \mathbf{X}' \rangle$

- 1: $(R_2, \dots, R_n) \leftarrow \text{rand}(\mathcal{K}^{n-1})$
 - 2: $R_1 \leftarrow \sum_{i=2}^n R_i \cdot L_i$;
 - 3: $\mathbf{X}' = \mathbf{X} + \mathbf{R}$;
-

Algorithm 4.5 Refresh vector: $\mathbf{Y} \leftarrow \text{SecIPRefresh}_{\mathbf{L}}(\mathbf{X})$

Input: vector \mathbf{X}
Output: vector \mathbf{Y} such that $\langle \mathbf{L}, \mathbf{X} \rangle = \langle \mathbf{L}, \mathbf{Y} \rangle$

- 1: $\mathbf{Y}_0 = \mathbf{X}$;
 - 2: **for** $i = 1$ to n **do**
 - 3: $\mathbf{Y}_i = \text{IPRefresh}_{\mathbf{L}}(\mathbf{Y}_{i-1})$;
 - 4: $\mathbf{Y} = \mathbf{Y}_n$;
-

- For each observation as in group (3), X_i can be perfectly simulated, $R_{i,j}$ can be sampled as in the real execution of the algorithm, and then all the partial sums $Y_{i,j}$ can be computed.

As for the output wires, we distinguish two cases. If some partial sum has already been observed, we remark that each output share $Y_{i,n}$ involves the computation of $n-1$ random bits $R_{i,1}, \dots, R_{i,n-1}$. The situation can be better understood from the following matrix, which shows the use of the random bits for each output share.

$$\begin{pmatrix}
 R_{1,1} & R_{1,2} & \dots & R_{1,n-1} & (\sum_{k=1}^{n-1} R_{1,k} L_k) L_n^{-1} & \leftarrow & Y_{1,n} \\
 R_{2,1} & R_{2,2} & \dots & R_{2,n-1} & (\sum_{k=1}^{n-1} R_{2,k} L_k) L_n^{-1} & \leftarrow & Y_{2,n} \\
 \vdots & \vdots & & \vdots & \vdots & & \\
 R_{n,1} & R_{n,2} & \dots & R_{n,n-1} & (\sum_{k=1}^{n-1} R_{n,k} L_k) L_n^{-1} & \leftarrow & Y_{n,n}
 \end{pmatrix}$$

Now, since the adversary can have just other $n-2$ observations, there exists at least one non-observed random bit and we can simulate $Y_{i,n}$ as a uniform and independent random value. Moreover, if all the partial sums have been observed, we can use the values previously simulated and add them according to the algorithm. Otherwise, if no partial sum has been probed, since the random values involved in the computation of $Y_{1,n}, \dots, Y_{i-1,n}, Y_{i+1,n}, \dots, Y_{n,n}$ are picked at random independently from that one of $Y_{i,n}$, we can again simulate $Y_{i,n}$ as a uniform and independent random value, completing the proof. \square

The following lemma shows the security of the multiplication scheme $\text{IPMult}_{\mathbf{L}}^{(1)}$.

Algorithm 4.6 Multiply masked values: $\mathbf{C} \leftarrow \text{IPMult}_{\mathbf{L}}^{(1)}(\mathbf{A}, \mathbf{B})$

Input: vectors \mathbf{A} and \mathbf{B} of length n

Output: vector \mathbf{C} such that $\langle \mathbf{L}, \mathbf{C} \rangle = \langle \mathbf{L}, \mathbf{A} \rangle \cdot \langle \mathbf{L}, \mathbf{B} \rangle$

```

1:  ▷ Computation of the matrix  $\mathbf{T}$ 
2:  for  $i = 1$  to  $n$  do
3:    for  $j = 1$  to  $n$  do
4:       $T_{i,j} = A_i \cdot B_j \cdot L_j$ ;
5:  ▷ Computation of the matrices  $\mathbf{U}$  and  $\mathbf{U}'$ 
6:  for  $i = 1$  to  $n$  do
7:    for  $j = 1$  to  $n$  do
8:      if  $i < j$  then
9:         $U'_{ij} \leftarrow \text{rand}(\mathcal{K})$ ;
10:     if  $i > j$  then
11:        $U'_{i,j} = -U'_{j,i}$ ;
12:      $U_{i,j} = U'_{i,j} \cdot \delta_{ij} L_i^{-1}$ ;
13:  ▷ Computation of the matrix  $\mathbf{V}$ 
14:   $\mathbf{V} = \mathbf{T} + \mathbf{U}$ ;
15:  ▷ Computation of the output vector  $\mathbf{C}$ 
16:  for  $i = 1$  to  $n$  do
17:     $C_i = \sum_j V_{i,j}$ ;

```

Lemma 4.3. *The algorithm $\text{IPMult}_{\mathbf{L}}^{(1)}$ is $t - \text{SNI}$ with $t = n - 1$.*

Proof. Let $\Omega = (\mathcal{I}, \mathcal{O})$ be a set of t observations respectively on the internal and on the output wires, where $|\mathcal{I}| = t_1$ and in particular $t_1 + |\mathcal{O}| \leq t$. We construct a perfect simulator of the adversary's probes, which makes use of at most t_1 shares of the secrets \mathbf{A} and \mathbf{B} .

Let w_1, \dots, w_t be the probed wires. We classify the internal wires in the following groups:

- (1) A_i, B_i ,
- (2) $U_{i,j}, U'_{i,j}$,
- (3) $A_i \cdot B_j, T_{i,j}, V_{i,j}$,
- (4) $C_{i,j}$, which represents the value of C_i at iteration i, j of the last **for** loop.

We define two sets of indices I and J such that $|I| \leq t_1$, $|J| \leq t_1$ and the values of the wires w_h with $h = 1, \dots, t$ can be perfectly simulated given only the knowledge of $(A_i)_{i \in I}$ and $(B_i)_{i \in J}$. The sets are constructed as follows.

- Initially I and J are empty.
- For every wire as in groups (1), (2) and (4), add i to I and to J .

- For every wire as in group (3) if $i \notin I$ add i to I and if $j \notin J$ add j to J .

Since the adversary is allowed to make at most t_1 internal probes, we have $|I| \leq t_1$ and $|J| \leq t_1$.

We now show how the simulator behaves, by starting to consider the internal observed wires.

1. For each observation as in group (1), by definition of I and J the simulator has access to A_i and B_i , and therefore the values are perfectly simulated.
2. For each observation as in group (2), we distinguish two possible cases:
 - If $i \in I, J$ and $j \notin J$, the simulator assigns a random and independent value to $U'_{i,j}$: if $i < j$ this is what would happen in the real algorithm, otherwise since $j \notin J$ the element U'_{ij} will never enter into the computation of any w_h (otherwise j would be in J).
 - If $i \in I, J$ and $j \in J$, the values $U'_{i,j}$ and $U'_{j,i}$ can be computed as in the actual algorithm: one of them (say $U'_{j,i}$) is assigned to a random and independent value and the other $U'_{i,j}$ to $-U'_{j,i}$.

The value $U_{i,j}$ is computed using the simulated $U'_{i,j}$ and the public value L_i .

3. For each observation as in group (3), by definition of the sets I and J and from the previous points, the simulator has access to A_i, A_j, B_i, B_j , to the public value L_j and $U_{i,j}, U'_{i,j}$ can be simulated. Therefore $A_i \cdot B_j, T_{i,j}$ and $V_{i,j}$ can be computed as in the real algorithm.
4. For each observation as in group (4), by definition $i \in I, J$. At first we assign a random value to every summand V_{ik} , with $k \leq j$ and $k \notin J$, entering in the computation of any observed C_{ij} . Then if one of the addends V_{ik} with $k \leq j$ composing C_{ij} has been probed, since by definition $k \in J$, we can simulate it as in Step 3. Otherwise V_{ik} has been previously assigned at the beginning of the current Step 4.

We now simulate the output wires C_i . We have to take into account two cases. At first, the case when the attacker has already observed only a partial sum C_{ij} . In this case the remaining part which has not been simulated is, for $1 < k \leq n$, of the form $\sum_{j=k}^n V_{i,j} = \sum_{j=k}^n (T_{i,j} + U_{i,j})$ if $k > i$ and $\sum_{j=k}^n V_{i,j} = \sum_{j=k,j \neq i}^n (T_{i,j} + U_{i,j}) + T_{i,i}$ otherwise. We note that each C_i depends on the random values in the i^{th} row of the matrix \mathbf{U}' , i.e. U'_{il} for $l < i$ and U'_{li} for $l > i$. In particular each of the U'_{il} appears a second time in one of the remaining $C_1, \dots, C_{i-1}, C_{i+1}, \dots, C_n$, as shown in the following matrix.

$$\left(\begin{array}{ccccc} 0 & U'_{1,2} & U'_{1,3} & \dots & U'_{1,n} \\ -U'_{1,2} & 0 & U'_{2,3} & \dots & U'_{2,n} \\ -U'_{1,3} & -U'_{2,3} & 0 & \dots & U'_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -U'_{1,n} & -U'_{2,n} & -U'_{3,n} & \dots & 0 \end{array} \right) \begin{array}{l} \leftarrow C_1 \\ \leftarrow C_2 \\ \\ \\ \leftarrow C_n \end{array}$$

Since each C_i depends on $n - 1$ random values and the adversary may have probed at most $n - 2$ of that, then, independently of the intermediate elements probed, at least one of the U'_{il} does not enter into the computation of C_{ij} and so C_i can be simulated as a random value. Moreover, if all the partial sums have been observed, we can use the values previously simulated and add them according to the algorithm. Finally it remains to simulate a C_i when no partial sum C_{ij} has been observed. By definition, at least one of the U'_{il} involved in the computation of C_i is not used in any other observed wire. Therefore we can assign a random value to C_i and complete the proof. \square

4.2. Application to AES S-box

Since $\text{IPMult}_{\mathbf{L}}^{(1)}$ is proved to be $t - \text{SNI}$, it can easily be composed with other gadgets, keeping the probing security. In this section, we analyze in detail the algorithm for the exponentiation to the power 254 in $\text{GF}(2^8)$, i.e., the non-linear part of the AES S-box. We consider Rivain and Prouff's algorithms chain from [Cor+13; RP10], depicted in Figure 2.6.

We recall the squaring routine $\text{IPSquare}_{\mathbf{L}}$ from [BFG15] in Algorithm 4.7. Considering that the multiplication gadget $\text{IPMult}_{\mathbf{L}}^{(1)}$ and the refreshing scheme $\text{SecIPRefresh}_{\mathbf{L}}$ are both $t - \text{SNI}$, and since the exponentiations $.^2, .^4$ and $.^{16}$ are linear functions in $\text{GF}(2^8)$, we can claim that the entire algorithm for the computation of $.^{254}$, using our schemes for IP masking, is $t - \text{SNI}$, according to the arguments of the security proof given in [Bar+15a].

4.2.1. A more efficient scheme for dependent inputs

We underline that for achieving $(n - 1)^{\text{th}}$ -order security the masked inputs \mathbf{A} and \mathbf{B} of $\text{IPMult}_{\mathbf{L}}^{(1)}$ must be mutually independent. If this is not the case, a refreshing of one of the factors is needed before processing the multiplication.

In this section we present an extended multiplication scheme $\text{IPMult}_{\mathbf{L}}^{(2)}$, illustrated in Algorithm 4.8, which can securely receive as input two values of the form \mathbf{A} and $g(\mathbf{A})$, where g is a linear function. Thanks to this property, in case of mutual dependence of the inputs, the refreshing is no longer needed and we can save on the number of random

Algorithm 4.7 Square masked variable: $\mathbf{Y} \leftarrow \text{IPSquare}_{\mathbf{L}}(\mathbf{X})$

Input: vector \mathbf{X}

Output: vector \mathbf{Y} such that $\langle \mathbf{L}, \mathbf{Y} \rangle = \langle \mathbf{L}, \mathbf{X} \rangle \cdot \langle \mathbf{L}, \mathbf{X} \rangle$

- 1: **for** $i = 1$ to n **do**
 - 2: $Y_i \leftarrow (X_i)^2 \cdot L_i;$
-

bits. The main idea of the new algorithm is to introduce a vector \mathbf{u} sampled at random at the beginning of the execution and used to internally refresh the shares of the secrets.

The correctness of $\text{IPMult}_{\mathbf{L}}^{(2)}$ is proved in the following.

Lemma 4.4 (Correctness). *For any $\mathbf{L}, \mathbf{A} \in \mathcal{K}^n$ and $\mathbf{C} = \text{IPMult}_{\mathbf{L}}^{(2)}(\mathbf{A}, g(\mathbf{A}))$, we have*

$$\langle \mathbf{L}, \mathbf{C} \rangle = \langle \mathbf{L}, \mathbf{A} \rangle \cdot \langle \mathbf{L}, g(\mathbf{A}) \rangle.$$

Proof. First notice that for all $i \neq j$ the following relation holds:

$$\begin{aligned} V_{i,j} &= (T_{i,j} + U_{i,j}) - B'_j = A'_{i,j} \cdot g(A_j) \cdot L_j + U'_{i,j} \cdot L_i^{-1} - g(A_j) \cdot u_j \cdot L_j \\ &= A_i \cdot g(A_j) \cdot L_j + u_j \cdot g(A_j) \cdot L_j + U'_{i,j} \cdot L_i^{-1} - g(A_j) \cdot u_j \cdot L_j \\ &= A_i \cdot g(A_j) \cdot L_j + U'_{i,j} \cdot L_i^{-1} \end{aligned}$$

Using the above, we get:

$$\begin{aligned} \langle \mathbf{L}, \mathbf{C} \rangle &= \sum_i L_i \cdot C_i = \sum_i L_i \sum_j V_{i,j} = \sum_i L_i \left(\sum_j (A_i \cdot g(A_j) L_j + U'_{i,j} \cdot L_i^{-1}) \right) \\ &= \sum_{i,j} (L_i \cdot A_i \cdot g(A_j) \cdot L_j + U'_{i,j}) = \sum_{i,j,i < j} (L_i \cdot A_i \cdot g(A_j) \cdot L_j + U'_{i,j}) \\ &\quad + \sum_{i,j,i > j} (L_i \cdot A_i \cdot g(A_j) \cdot L_j - U'_{i,j}) + \sum_i L_i \cdot A_i \cdot g(A_i) \cdot L_i \\ &= \sum_{i,j} L_i \cdot A_i \cdot g(A_j) \cdot L_j = \sum_i L_i \cdot A_i \left(\sum_j g(A_j) \cdot L_j \right) = \langle \mathbf{L}, \mathbf{A} \rangle \cdot \langle \mathbf{L}, g(\mathbf{A}) \rangle \end{aligned}$$

□

Lemma 4.5 provides the security analysis of $\text{IPMult}_{\mathbf{L}}^{(2)}$.

Lemma 4.5. *Let g be a linear function over \mathcal{K} . The algorithm $\text{IPMult}_{\mathbf{L}}^{(2)}(\mathbf{A}, g(\mathbf{A}))$ is $t - \text{SNI}$, with $t = n - 1$.*

Proof. Let $\Omega = (\mathcal{I}, \mathcal{O})$ be a set of t observations respectively on the internal and on the output wires, where $|\mathcal{I}| = t_1$ and in particular $t_1 + |\mathcal{O}| \leq t$. We point out the existence of a perfect simulator of the adversary's probes, which makes use of at most t_1 shares of the secret \mathbf{A} .

Let w_1, \dots, w_t be the probed wires. We classify the internal wires in the following groups:

- (1) $A_i, g(A_i), g(A_i) \cdot u_i, B'_i, u_i$
- (2) $U_{i,j}, U'_{i,j}$
- (3) $A'_{i,j}, A'_{i,j} \cdot g(A_j), T_{i,j}, T_{i,j} + U_{i,j}, V_{i,j}$
- (4) $C_{i,j}$, which represents the value of C_i at iteration i, j of the last **for**

Algorithm 4.8 Multiply dependent masked values: $\mathbf{C} \leftarrow \text{IPMult}_{\mathbf{L}}^{(2)}(\mathbf{A}, g(\mathbf{A}))$

Input: vector \mathbf{A} of length n

Output: vector \mathbf{C} satisfying $\langle \mathbf{L}, \mathbf{C} \rangle = \langle \mathbf{L}, \mathbf{A} \rangle \cdot \langle \mathbf{L}, g(\mathbf{A}) \rangle$, for g linear function

```

1:  ▷ Sampling at random of the vector  $\mathbf{u}$ 
2:  for  $i = 1$  to  $n$  do
3:     $u_i \leftarrow \text{rand}(\mathcal{K})$ ;
4:  ▷ Computation of the matrix  $\mathbf{A}'$ 
5:  for  $i = 1$  to  $n$  do
6:    for  $j = 1$  to  $n$  do
7:       $A'_{i,j} = A_i + \delta_{ij}u_j$ ;
8:  ▷ Computation of the vector  $\mathbf{B}'$ 
9:  for  $i = 1$  to  $n$  do
10:    $B'_i = g(A_i) \cdot u_i \cdot L_i$ ;
11:  ▷ Computation of the matrix  $\mathbf{T}$ 
12:  for  $i = 1$  to  $n$  do
13:    for  $j = 1$  to  $n$  do
14:       $T_{i,j} = A'_{i,j} \cdot g(A_j) \cdot L_j$ ;
15:  ▷ Computation of the matrices  $\mathbf{U}$  and  $\mathbf{U}'$ 
16:  for  $i = 1$  to  $n$  do
17:    for  $j = 1$  to  $n$  do
18:      if  $i < j$  then
19:         $U'_{ij} \leftarrow \text{rand}(\mathcal{K})$ ;
20:      if  $i > j$  then
21:         $U'_{ij} = -U'_{ji}$ ;
22:       $U_{i,j} = U'_{i,j} \cdot \delta_{ij}L_i^{-1}$ ;
23:  ▷ Computation of the matrix  $\mathbf{V}$ 
24:  for  $i = 1$  to  $n$  do
25:    for  $j = 1$  to  $n$  do
26:       $V_{i,j} = (T_{i,j} + U_{i,j}) - \delta_{ij}B'_j$ ;
27:  ▷ Computation of the output vector  $\mathbf{C}$ 
28:  for  $i = 1$  to  $n$  do
29:     $C_i = \sum_j V_{i,j}$ ;

```

We now define the set of indices I with $|I| \leq t_1$ such that the wires w_h can be perfectly simulated given only the knowledge of $(A_i)_{i \in I}$. The procedure for constructing the set is the following:

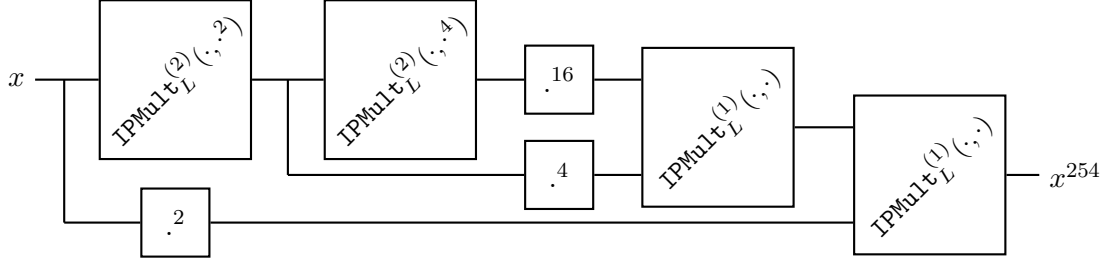
- Initially I is empty.
- For every wire as in the groups (1), (2) and (4), add i to I .
- For every wire as in the group (3), if $i \notin I$ add i to I and if $i \in I$ add j to I .

Since the adversary is allowed to make at most t_1 internal probes, we have that $|I| \leq t_1$.

In the simulation phase, at first we assign a random value to every u_i entering in the computation of any observed w_h . Then the simulation for all internal wires w_h proceeds as follows.

1. For each observation in category (1), then $i \in I$ and by definition we can directly compute from A_i, u_i and the public value L_i .
2. For each observation in category (2), then $i \in I$ and we distinguish two possible cases:
 - If $j \notin I$, then we can assign a random and independent value to $U'_{i,j}$. Indeed if $i < j$ this is what would happen in the real execution of the algorithm and if $i > j$, since $j \notin I$, $U'_{i,j}$ will never be used in the computation of other observed values. We compute $U_{i,j}$ using $U'_{i,j}$ and the public value L_i .
 - If $j \in I$, the values $U'_{i,j}$ and $U'_{j,i}$ can be computed as in the actual circuit: we assign one of them (say $U'_{j,i}$) to a random and independent value and the other $U'_{i,j}$ to $-U'_{j,i}$. We compute $U_{i,j}$ using $U'_{i,j}$ and the public value L_i .
3. For each observation in category (3), then $i \in I$ and we distinguish two possible cases:
 - If $j \notin I$, then we can assign a random and independent value to w_h . Indeed, since $j \notin I$, one of the values composing w_h has not been observed (otherwise by construction j would be in I) and for the same reason also any of the w_h does not enter in the expression of any other observed wire.
 - If $j \in I$, the value w_h can be perfectly simulated by using the accessible values $A_i, g(A_j), u_i, u_j, L_i, L_j$ and the values $U_{i,j}, U'_{i,j}$ assigned in Step 2.
4. For each observation as in the group (4), by definition $i \in I$. At first we assign a random value to every summand V_{ik} , with $k \leq j$ and $k \notin I$, entering in the computation of any observed C_{ij} . Then if one of the addends V_{ik} with $k \leq j$ composing C_{ij} has been probed, since by definition $k \in I$, we can simulate it as in Step 3. Otherwise V_{ik} has been previously assigned at the beginning of the current Step 4.

As for the probed output wires, if the attacker has already observed a partial sum C_{ij} , for $1 < k \leq n$ the remaining non-simulated part is $\sum_{j=k}^n V_{i,j} = \sum_{j=k}^n (T_{i,j} + U_{i,j}) - B'_j$ if $k > i$ and $\sum_{j=k}^n V_{i,j} = \sum_{j=k, j \neq i}^n (T_{i,j} + U_{i,j}) - B'_j + T_{i,i}$ otherwise. Using a similar argument to the one in the proof of Lemma 4.3, we point out that C_i can be simulated as a random value. Moreover, if all the partial sums have been observed, we can use the values


 Figure 4.4.: Gadget $.^{254}$ which makes use of $\text{IPMult}_{\mathbf{L}}^{(1)}$ and $\text{IPMult}_{\mathbf{L}}^{(2)}$

previously simulated and add them according to the algorithm. Finally, when no partial sum C_{ij} has been observed, again as before, by definition at least one of the U'_{il} involved in the computation of C_i is not used in any other observed wire and then we can assign to C_i a random value, completing the proof. \square

We can now exploit this new scheme in the $.^{254}$ algorithm, by eliminating the first two refreshing and substituting the first two multiplications with our $\text{IPMult}_{\mathbf{L}}^{(2)}(., .^2)$ and $\text{IPMult}_{\mathbf{L}}^{(2)}(., .^4)$, while using in the rest the $\text{IPMult}_{\mathbf{L}}^{(1)}$. In particular, according to the squaring routine in Algorithm 4.7, we point out that in $\text{IPMult}_{\mathbf{L}}^{(2)}(., .^2)$ the shares $g(A_i)$ correspond to the products $A_i^2 \cdot L_i$ and in $\text{IPMult}_{\mathbf{L}}^{(2)}(., .^4)$ the shares $g(A_i)$ correspond to the products $A_i^4 \cdot L_i \cdot L_i \cdot L_i$. The implementation of the gadget $.^{254}$ is depicted in Figure 4.4 and in Lemma 4.6 we prove that it is $t - \text{SNI}$, using the techniques presented in [Bar+15a].

Lemma 4.6. *Gadget $.^{254}$, shown in Figure 4.4, is $t - \text{SNI}$.*

Proof. Let $\Omega = (\bigcup_{i=1}^7 \mathcal{I}^i, \mathcal{O})$ a set of t observations respectively on the internal and output wires, where each \mathcal{I}^i refers to the gadget G^i and $\sum_{i=1}^7 |\mathcal{I}^i| + |\mathcal{O}| \leq t$. In the following we construct a simulator which makes use of at most $\sum_{i=1}^7 |\mathcal{I}^i|$ shares of the secret, by simulating each gadget in turn.

Gadget G^1 : Since $\text{IPMult}_{\mathbf{L}}^{(1)}$ is $t - \text{SNI}$ and $|\mathcal{I}^1 \cup \mathcal{O}| \leq t$, then there exist two sets of indices $\mathcal{S}_1^1, \mathcal{S}_2^1$ such that $|\mathcal{S}_1^1| \leq |\mathcal{I}^1|$, $|\mathcal{S}_2^1| \leq |\mathcal{I}^1|$ and the gadget can be perfectly simulated from its input shares corresponding to the indices in \mathcal{S}_1^1 and \mathcal{S}_2^1 .

Gadget G^2 : Since $\text{IPMult}_{\mathbf{L}}^{(1)}$ is $t - \text{SNI}$ and $|\mathcal{I}^2 \cup \mathcal{S}_2^1| \leq |\mathcal{I}^1| + |\mathcal{I}^2| \leq t$, then there exist two sets of indices $\mathcal{S}_1^2, \mathcal{S}_2^2$ such that $|\mathcal{S}_1^2| \leq |\mathcal{I}^2|$, $|\mathcal{S}_2^2| \leq |\mathcal{I}^2|$ and the gadget can be perfectly simulated from its input shares corresponding to the indices in \mathcal{S}_1^2 and \mathcal{S}_2^2 .

Gadget G^3 : Since $.^{16}$ is affine, there exists a set of indices \mathcal{S}^3 such that its cardinality is $|\mathcal{S}^3| \leq |\mathcal{I}^3| + |\mathcal{S}_2^2|$ and the gadget can be perfectly simulated from its input shares corresponding to the indices in \mathcal{S}^3 .

	#ADDITIONS	# MULTIPLICATIONS	#RANDOMS
$\text{IPMult}_{\mathbf{L}}^{(1)}$	$2n^2$	$3n^2$	$\frac{n(n-1)}{2}$
$\text{IPMult}_{\mathbf{L}}^{(2)}$	$4n^2$	$3n^2 + 2n$	$\frac{n(n+1)}{2}$
$\text{SecIPRefresh}_{\mathbf{L}}$	$2n^2 - n$	$2n$	n^2
$\text{IPMult}_{\mathbf{L}}^{(1)}$ and $\text{SecIPRefresh}_{\mathbf{L}}$	$4n^2 - n$	$3n^2 + 2n$	$\frac{n(3n-1)}{2}$
IPMult in [BFG15]	$(n+1)(2n-1)+1$	$n(3n+1)$	n^2
Alg. 5 in [Cor+13]	$4n(n-1)$	–	$n(n-1)$
Alg. 3 in [Bel+16]	$4n(n-1)$	$\frac{1}{4}(n-1)(7n+3)$ (n odd) $\frac{1}{4}n(7n-6)$ (n even)	$n(n-1)$

Table 4.1.: Complexity of $\text{IPMult}_{\mathbf{L}}^{(1)}$ and $\text{IPMult}_{\mathbf{L}}^{(2)}$ and comparison with the IPMult in [BFG15] and the multiplication algorithms of [Cor+13] and [Bel+16]

Gadget G^4 : Since \cdot^4 is affine, there exists a set of indices \mathcal{S}^4 such that $|\mathcal{S}^4| \leq |\mathcal{I}^4| + |\mathcal{S}_1^2|$ and the gadget can be perfectly simulated from its input shares corresponding to the indices in \mathcal{S}^4 .

Gadget G^5 : Since $\text{IPMult}_{\mathbf{L}}^{(2)}$ is t -SNI and $|\mathcal{I}^5 \cup \mathcal{S}^3| \leq |\mathcal{I}^5| + |\mathcal{I}^3| + |\mathcal{I}^2| \leq t$, then there exists a set of indices \mathcal{S}^5 such that $|\mathcal{S}^5| \leq |\mathcal{I}^5|$ and the gadget can be perfectly simulated from its input shares corresponding to the indices in \mathcal{S}^5 .

Gadget G^6 : Since $\text{IPMult}_{\mathbf{L}}^{(2)}$ is t -SNI and $|\mathcal{I}^6 \cup \mathcal{S}^5| \leq |\mathcal{I}^6| + |\mathcal{I}^5| \leq t$, then there exists a set of indices \mathcal{S}^6 such that $|\mathcal{S}^6| \leq |\mathcal{I}^6|$ and the gadget can be perfectly simulated from its input shares corresponding to the indices in \mathcal{S}^6 .

Gadget G^7 : Since \cdot^2 is affine, there exists a set of indices \mathcal{S}^7 such that its cardinality is $|\mathcal{S}^7| \leq |\mathcal{I}^7| + |\mathcal{S}_1^1| \leq |\mathcal{I}^7| + |\mathcal{I}^1|$ and the gadget can be perfectly simulated from its input shares corresponding to the indices in \mathcal{S}^7 .

Each of the previous steps guarantees the existence of a simulator for the respective gadgets. The composition of them allows us to construct a simulator of the entire circuit which uses $\mathcal{S}^6 \cup \mathcal{S}^7$ shares of the input. Since $|\mathcal{S}^6 \cup \mathcal{S}^7| \leq |\mathcal{I}^7| + |\mathcal{I}^1| + |\mathcal{I}^6| \leq \sum_{i=1}^7 |\mathcal{I}^i|$ we can conclude that the gadget \cdot^{254} is t -SNI. \square

The advantage of the use of $\text{IPMult}_{\mathbf{L}}^{(2)}$ mostly consists in amortizing the randomness complexity. Indeed the new scheme requires only n (for the vector \mathbf{u}) plus $\frac{n(n-1)}{2}$ (for the matrix \mathbf{U}) random bits, while the previous one uses a larger amount of randomness, corresponding to n^2 (for the $\text{SecIPRefresh}_{\mathbf{L}}$) plus $\frac{n(n-1)}{2}$ (for the $\text{IPMult}_{\mathbf{L}}^{(1)}$) bits. We summarize in Table 4.1 the complexities of the two schemes.

The issue of providing a secure multiplication of two dependent operands was first addressed by Coron *et al.* in [Cor+13]. In their work the authors proposed a new algorithm which requires $n(n-1)$ random bits and that has later been proved to be $t - \text{SNI}$ in [Bar+15a]. By analyzing the amount of random generations and comparing it to the one of $\text{IPMult}_{\mathbf{L}}^{(2)}$, we can see that our scheme is more efficient whenever $n > 3$, while it requires the same amount of randomness for $n = 3$ and more random bits for $n < 3$. On the other hand, from a complexity point of view the scheme in [Cor+13] is better optimized in terms of field multiplications since it makes use of look-up tables.

A more detailed performance analysis is provided in [Bal+17] and summarized in the next section.

4.3. Performance evaluations

In this section, we briefly summarize the results of the performance evaluations of our improved IP masking construction performed in [Bal+17].

The authors developed masked software implementations of the AES-128 encryption for AVR architectures, using either our new multiplication scheme $\text{IPMult}_{\mathbf{L}}^{(1)}$ alone, or in combination with $\text{IPMult}_{\mathbf{L}}^{(2)}$. Additionally, they also developed protected instances of AES-128 with Boolean masking, in order to compare performances.

The algorithm considered for the power function x^{254} is the one in [Cor+13; RP10] represented in Figure 2.6. Additionally, they provided a faster implementation using the addition chain proposed by Grosso *et al.* [GPS14], which leverages on the algorithm introduced by Coron *et al.* [Cor+13] to securely evaluate functions of the form $x \cdot g(x)$, where g is a linear function.

We report in Table 4.2 the performance evaluations for assembly implementations with $n = 2, 3$ shares. The implementation protected by IP masking using only $\text{IPMult}_{\mathbf{L}}^{(1)}$ represents a notable improvement compared to earlier work [BFG15]. Our implementation requires roughly 157 k cycles and 372 k cycles for security levels $n = 2$ and $n = 3$, respectively, against the 375 k and 815 k cycles needed to protect instances of AES-128 for the same security levels in [BFG15]. The implementation protected by IP masking using $\text{IPMult}_{\mathbf{L}}^{(2)}$ in combination with $\text{IPMult}_{\mathbf{L}}^{(1)}$ achieves weaker performance in terms of cycles but, as mentioned earlier, it improves the randomness complexity. The results for Boolean masking are 110 k and 230 k, respectively. We point out that the source of the timing gap for IP masking is the computation of x^{254} , since the rest of AES operations perform in a similar amount of cycles. IP masking is slower than the Boolean masking, mainly due to the extra operations to perform in the multiplication gadgets. However, since \mathbf{L} is fixed, it is possible to optimize the cycle count by tabulating the field multiplications with elements L_i and L_i^{-1} . This technique costs more in terms of non-volatile storage and it is feasible only for small numbers of shares n . The results of the

MASKING		TIMINGS		MEMORY
		x^{254}	AES-128	
IP masking (only $\text{IPMult}_{\mathbf{L}}^{(1)}$)	n=2	709	157 196	2 816
	n=3	1 752	372 225	3 328
IP masking ($\text{IPMult}_{\mathbf{L}}^{(1)}$ and $\text{IPMult}_{\mathbf{L}}^{(2)}$)	n=2	763	167 996	2 816
	n=3	1 766	375 025	3 328
Boolean masking (addition chain [RP10])	n=2	459	110 569	2 048
	n=3	1 043	230 221	2 048
Boolean masking (addition chain [GPS14])	n=2	275	73 769	1 792
	n=3	676	160 357	1 792

Table 4.2.: Performance evaluation of protected AES-128 implementations on AVR architectures (optimized in assembly code). Timings in clock cycles, memory requirements in bytes [Bal+17]

evaluations show a decrease in the gap between Boolean and IP masking implementations to slightly more than a factor 2.

4.4. Information theoretic evaluation

In this section, we extend our investigations by reporting on the information theoretic evaluation of the Inner Product encoding, which was conducted in [Bal+17].

Linear (e.g., Hamming weight) leakages

We start by reporting on the results of the analysis of the information leakage of the Inner Product encoding of Algorithm 4.2 for $n = 2$ shares and a Hamming weight leakage function. We represent in Figure 4.5 the result of the information theoretic analysis for Hamming weight leakages, for vectors $L_2 = 17, 5, 7$ and noise variances between 10^{-2} and 10^3 . In the figure, it is additionally reported the leakage of an unprotected target secret variable A and of a Boolean encoding (which is a special case of Inner Product encoding such that $L_1 = L_2 = 1$) for illustration. We can observe that, for low noise levels, the increased algebraic complexity of Inner Product masking allows significantly lower leakages than Boolean masking, as already discussed in [Bal+12; BFG15]. On an intuitive level, the reason of such phenomenon can be explained by the fact that, in Boolean masking, knowing one bit of each share leads directly to one bit of the secret, while, in Inner Product masking, it only leads to a (smaller) bias on the secret variable distribution.

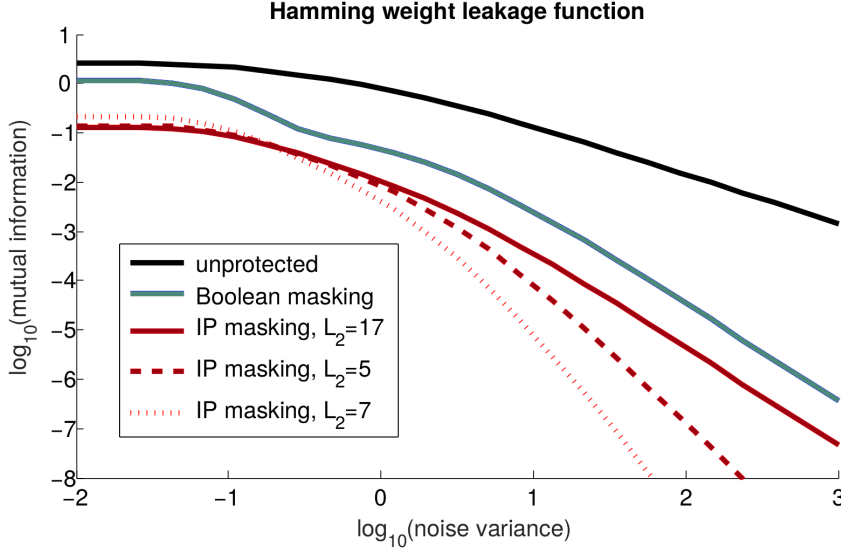


Figure 4.5.: Information theoretic evaluation of an Inner Product encoding with Hamming weight leakage function [Bal+17]

For large noise levels, we recognize the security order (in the bounded moment model) of the masking schemes according to the slope of the information theoretic curves. The latter one moves from -1 for an unprotected implementation to -2 for Boolean masking, corresponding to a security order 1 in the bounded moment model. Moreover, by considering different values of L_2 in the Inner Product encoding, the results also show that the slope of the information theoretic curves can be reduced. In particular, it reaches a level of -3 (which corresponds to a security order 2 in the bounded moment model) and even -4 (which corresponds to a security order 3 in the bounded moment model). The reasons of such order amplifications were analyzed in [Wan+16] and [Bal+17], and we refer to these works for further details.

Non-linear (e.g., random) leakages

Let us now consider the case when, in place of the Hamming weight leakages, we have a random leakage function G with similar output range $\{0, 1, \dots, 8\}$.

The result of the information theoretic analysis for random leakages, noise variances between 10^{-2} and 10^3 , and vectors $L_2 = 17, 5, 7$ is given in Figure 4.5, where it is again reported the leakage of an unprotected A and a Boolean encoding. We can see that, for large noise levels, the security order amplification vanishes and all the information

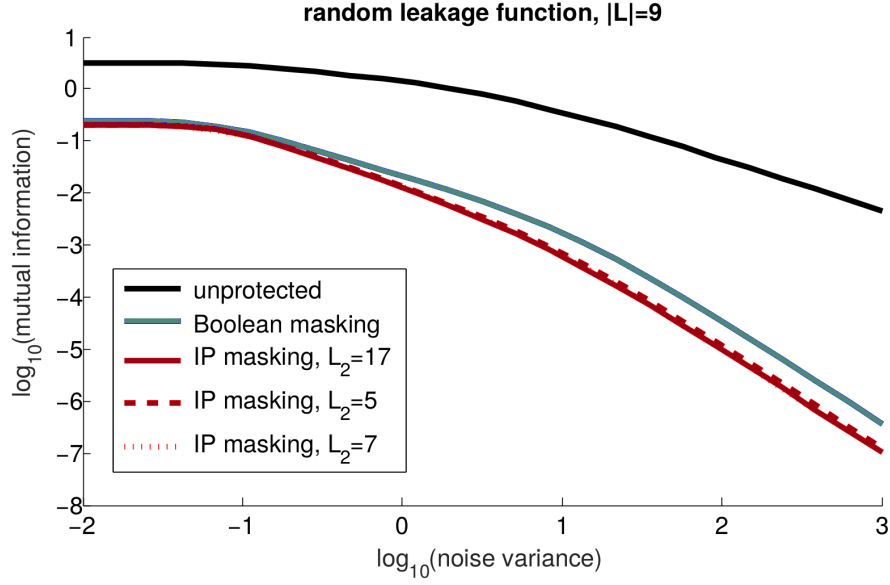


Figure 4.6.: Information theoretic evaluation of an Inner Product encoding with random leakage function [Bal+17]

theoretic curves corresponding to $d = 2$ shares have slope -2 , as predicted by the proofs in the probing model. This phenomenon is due to the non-linear leakage function, that recombines the shares, reducing the security order. Additionally, we notice that making the leakage function non-linear compensates the low algebraic complexity of the Boolean encoding and therefore reduces the distance between Boolean and Inner Product encodings. These experiments show that the security order amplification of Inner Product masking is highly implementation-dependent.

4.5. Empirical side-channel leakage evaluation

We complete our analysis by providing the results of empirical side-channel leakage evaluations for both Boolean and IP masking with two shares, and $L_1 = 1, L_2 = 7$, conducted in [Bal+17].

The target platform is an STM32 Nucleo board equipped with an ARM Cortex-M4 processor core. The processor runs at 168 MHz and features a built-in RNG capable of generating 32-bit random numbers every 40 clock cycles.

RNG deactivated

We first show the evaluation of both implementations with the RNG deactivated, i.e., when all random numbers are zero.

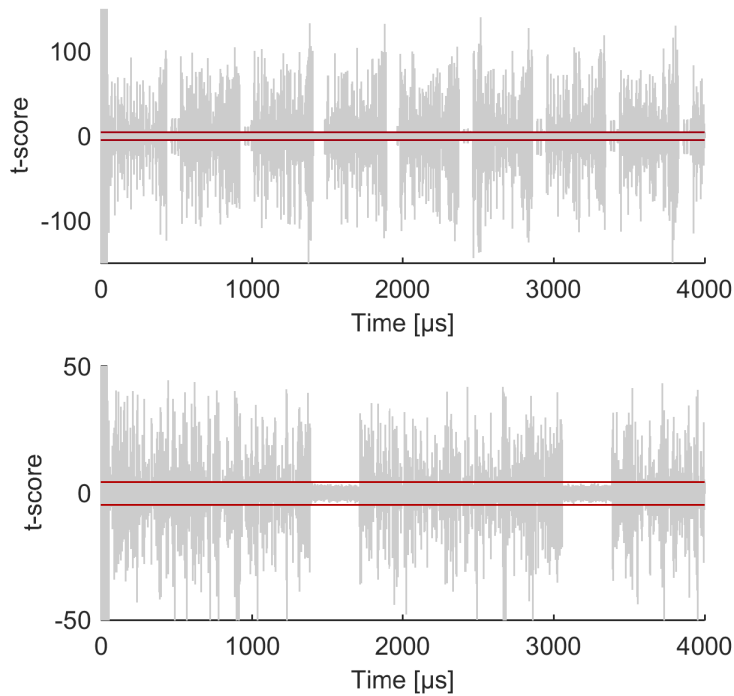


Figure 4.7.: t-test results for Boolean masking (top) and IP masking (bottom) with RNG deactivated; each based on 10000 measurements. The red lines mark the ± 4.5 threshold [Bal+17]

The authors consider 10000 measurements from each implementation. Figure 4.7 depicts the plots of the t-scores for Boolean masking and IP masking in gray (respectively at the top and bottom of the figure). The red lines indicate the ± 4.5 threshold, which is typically used in the literature and roughly corresponds to 99.999% confidence.

In this scenario, both implementations leak significantly, and repetitive patterns in the plots of the t-scores reveal the rounds of AES (the areas with high t-scores) interleaved by the key scheduling. However, even if both implementations leak, we can already notice that the implementation protected with IP masking has lower t-scores, corresponding to less evidence of leakage, compared to the Boolean ones.

RNG activated

Now we show the evaluation with activated RNG. This time the authors consider 1 million measurements. Figure 4.8 displays the results for both Boolean masking and IP masking (respectively at the top and at the bottom of the figure).

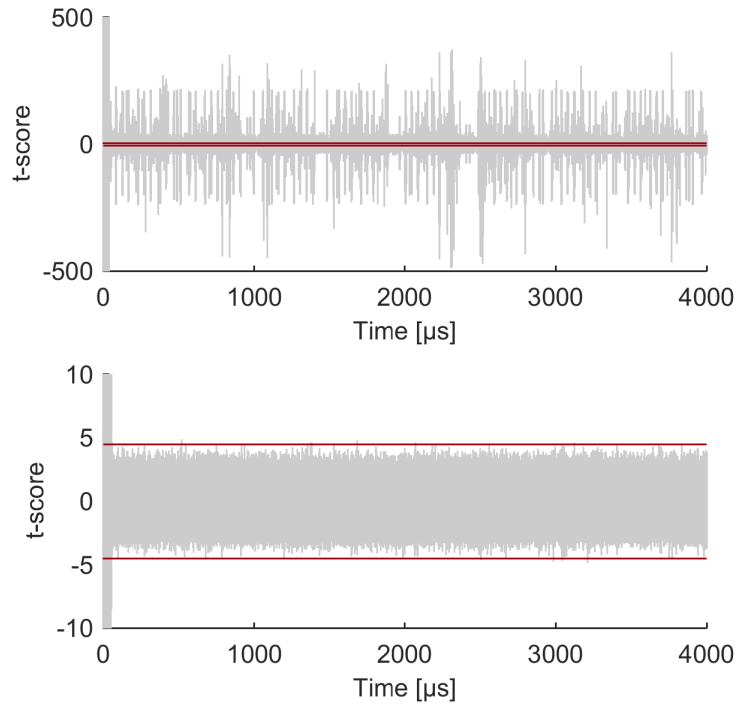


Figure 4.8.: t-test results for Boolean masking (top) and IP masking (bottom) with RNG activated; each based on 1 million measurements. The red lines mark the ± 4.5 threshold [Bal+17]

This time we can see a big difference between the two masking schemes. The implementation protected with Boolean masking leaks. On the other hand, the one protected with IP masking shows significantly less evidence of leakage, and, in particular, it does not show any significant leakage.

These results show that the more complicated algebraic structure of the Inner Product encoding leads to less evidence of leakage in practice compared to the Boolean masking.

4.6. Conclusions

In this chapter, we examined the Inner Product masking, which is characterized by a more complicated algebraic structure compared to Boolean masking. We showed that

this property leads to less evidence of leakage in practice, and, in some cases, to a security order amplification. However, such qualities come with the price of performance overhead.

We focused on such main drawback and suggested improvements over the previous Inner Product schemes. In particular, we proposed new multiplication algorithms that are conceptually close to the schemes of Ishai *et al.* [ISW03] and improve the efficiency of prior work. Compared to the previous schemes in [BFG15], our new algorithms show a reduced randomness cost, while providing the strong security property of $t - \text{SNI}$. Additionally, we provided a multiplication scheme optimized for the case of dependent inputs, which allow us to save on the cost of refreshing one of the inputs. Since this composition pattern is typical in the algorithm for the exponentiation x^{254} , such an optimization is particularly helpful to decrease the randomness complexity of Inner Product masked implementation of the AES.

These improvements in the performance overhead showed how the Inner Product masking could be a valid alternative to Boolean masking.

5. Masking hardware oriented implementations: the robust probing model

In this chapter, we move our attention to another research direction, which consists in discussing and formalizing the security requirements for a secure masked implementation in hardware.

Motivations

The analysis conducted until now in this work focused only on software-oriented masking, i.e., on masking schemes principally designed for software implementations. However, also the protection of hardware implementations against side-channel attacks is an important objective in cryptographic engineering. In this setting, one crucial additional issue is the presence of physical defaults, such as glitches (i.e., combinatorial recombinations) [MPG05; MPO05], transition-based leakages (i.e., memory recombinations) [Bal+14; Cor+12] and couplings (i.e., routing recombinations) [Cnu+17], that can provide extra information for each adversarial probe.

In the last years, multiple solutions have been proposed in the literature to extend software countermeasures to protect hardware implementations. Some examples are Threshold Implementations (TIs), Consolidated Masking (CMS) scheme in [Cnu+16; Rep+15], the Domain-Oriented Masking (DOM) in [GMK16; GMK17], the Unified Masking Approach (UMA) in [GM17] and the Generic Low Latency Masking (GLM) in [GIB18]. However, while provably security is at the base of software-oriented masking, the schemes deployed in the context of hardware-oriented masking have not been formally proven for arbitrary security orders so far, nor their composability guarantees have been discussed in detail. For instance, TI, which showed that the simple property of non-completeness³ is sufficient to counteract the glitch issue [NRS11], found some obstacle on the way to its generalization at high-order and the first attempt to build a higher-order TI in [Bil+14] failed due to a lack of refreshing, causing a composability flaw [Rep15; Rep+15].

When dealing with software implementations, avoiding such flaws is not a difficult task,

³See Section 2.7 for details.

as security proofs in the probing model of Ishai *et al.* [ISW03], according to the notions of Non-Interference (NI) and Strong Non-Interference (SNI), can be used to ensure the compositional security of masked gadgets [Bar+16].

Moreover, next to such theoretical approaches, security analysis can be supported by practical ones as well, exploiting program verification techniques. For instance, Barthe *et al.* [Bar+15b] developed a tool capable of verifying the security of a masked implementation up to a specific order. Other examples are the works in [EWS14; Rep16], which suggest similar but more specialized ideas. In practice, when in a software masked implementation security is claimed for arbitrary orders, we need to provide a hand-made security proof for the target gadgets. Additionally, in case the implementation considered is a full cipher combining several gadgets, the proof has to show that secure composability is ensured as well. On the other hand, if the security is claimed only up to a given order, program verification techniques can be used to analyze the target implementation.

In the intent of formalizing security requirements for hardware implementations, the properties that we desire to achieve are robustness against physical defaults and composability. Probing security is not enough for guaranteeing protection in this setting. Indeed, even if an implementation is proven not to leak sensitive information in the probing model, physical defaults can decrease the security order in the bounded moment model [Bar+17]. Specifically, the lowest key-dependent statistical moment of the leakage distribution is lower than the optimal value, which is the number of shares. In practice, we can explain this phenomenon in a recombination of the shares by the leakage function. On the other hand, a lack of composability can reduce the security order in the probing model as well, as shown in an attack by Coron *et al.* [Cor+13].

Tools that guarantee such properties of robustness and composability separately already exists. For instance, the security in the bounded moment model can be guaranteed by some algorithmic characteristics that moderate the shares' recombinations. An example already mentioned is TI, which offers an excellent technique to prevent glitches. The lazy engineering method in [Bal+14] is another example, where, by increasing the number of shares, full memory recombination can be ruled out. As for composability guarantees, the probing model provides formal methods that can ensure a secure combination of gadgets. However, no current model analyzes their combination.

In this context, we study in our work how to connect the security solutions discussed above and provide a unified model that simultaneously captures the issues concerning physical defaults and composability at any order, at the scope of guiding practically-oriented implementations.

Contribution

As a first contribution, we analyze the interesting case of first-order threshold implementations, which are characterized by their low randomness requirements. We explain

such a phenomenon by giving a slight variant of the concept of Strong Non-Interference (SNI) in Definition 2.5, which we call Pseudo-Strong Non-Interference. In this setting, we additionally discuss the natural tradeoff between the number of shares and the cycle count. In particular, we observe that the correctness property in threshold implementations does not necessarily have to be satisfied by the intermediate results, but it is enough that only the final result is correct.

In the second part of the chapter, we move to analyze the additional challenges raised by higher-order glitch-free implementations and provide a formal tool to analyze them. We introduce our main contribution, the *robust probing model*, which tweaks the original probing model in order to capture a broad class of physical defaults and can easily be combined with composability notions, as the (Strong) Non-Interference.

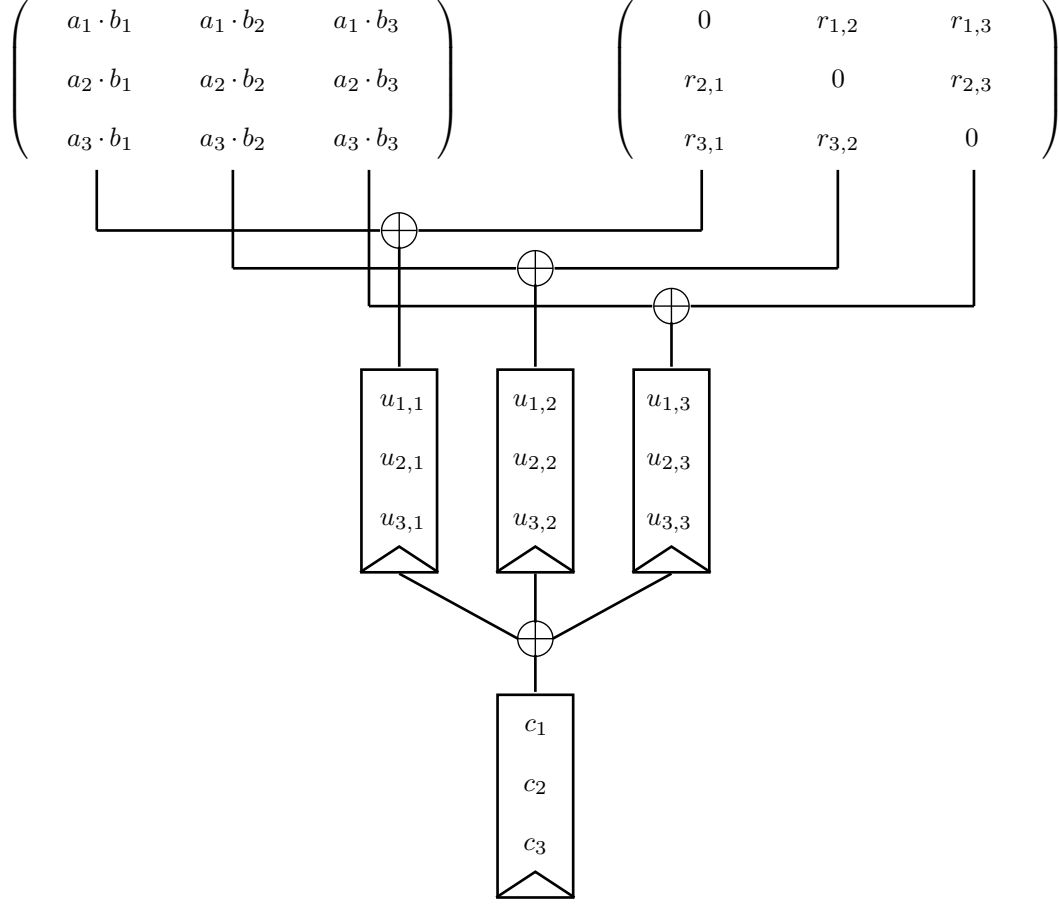
In order to capture the physical defaults of glitches, transitions and couplings, we define ϵ -extended probes. We distinguish them in specifically extended, when the model is dependent on the circuit topology, and generic extended when it is independent of it. Informally speaking (we refer to Section 5.2 for an in-depth discussion), according to the model for glitches, each observation is modeled as a specific ϵ -extended probe, where ϵ is equal to the number of input shares. In the context of transitions, each observation corresponds to a specific 2-extended probe. And as for couplings, each observation is modeled as a generic γ -extended probe, where γ is the number of couplings. Using such a new model, we discuss and conjecture simple propositions concerning the combination of physical defaults.

We then study concrete constructions of masked and threshold implementations, reaching the important conclusion that a 2-cycle implementation of Ishai *et al.*'s (slightly tweaked) multiplication algorithm [ISW03], or the parallel multiplication algorithm in [Bar+17], offers good robustness against glitches, while also offering good composability by design. In order to provide some intuition on the algorithm and on how the model is used more in detail, we illustrate the example of the 3-share implementation of the ISW multiplication scheme in 2 cycles, depicted in Figure 5.1, where the rectangles clearly indicate where to place the registers.

In the following, we give some basic ideas regarding the reason why the gadget is secure against glitches and explain how the extended probes look like in this context. Concretely, an adversary attacking the internal values of this scheme can observe the possible extended probes, each of them allowing to access between 2 and 3 shares:

- *1st stage:*

$$\begin{array}{lll}
p_{1,1} := (a_1, b_1, 0), & p_{1,2} := (a_1, b_2, r_{1,2}), & p_{1,3} := (a_1, b_3, r_{1,3}), \\
p_{2,1} := (a_2, b_1, r_{1,2}), & p_{2,2} := (a_2, b_2, 0), & p_{2,3} := (a_2, b_3, r_{2,3}), \\
p_{3,1} := (a_3, b_1, r_{1,3}), & p_{3,2} := (a_3, b_2, r_{2,3}), & p_{3,3} := (a_3, b_3, 0).
\end{array}$$


 Figure 5.1.: $(1,0,0)$ -robust 2-SNI implementation of ISW in 2 cycles

- *2nd stage:*

$$p_1 := (u_{1,1}, u_{1,2}, u_{1,3}), \quad p_2 := (u_{2,1}, u_{2,2}, u_{2,3}), \quad p_3 := (u_{3,1}, u_{3,2}, u_{3,3}).$$

Attacking output shares allows the observation of c_1, c_2, c_3 . Note that the probes on the output shares are standard ones, since the shares are stored in registers, preventing the propagation of glitches.

Moving to the simulation, we have to distinguish three possible options:

1. Both probes are on the internal shares (e.g., $p_{1,2}, p_1$).
2. One probe is internal, the other one is on the output shares (e.g., $p_{1,2}, c_1$).
3. Both probes are on the output shares (e.g., c_1, c_2).

In the first case, and according to the proof, we construct the set of indexes $I = \{1\}$ and $J = \{1, 2\}$. In the simulation phase we assign $r_{1,2}$ to a random value and we can

perfectly compute $p_{1,2}$ by having access to a_1 and b_2 . As for p_1 , we perfectly simulate the first component $u_{1,1}$ by using a_1 and b_1 ; since $2 \in J$ and $2 \notin I$ we can use the components of the probed $p_{1,2}$ to simulate the second component $u_{1,2}$; and since $3 \notin J$ we can pick a uniform and random value for simulating the third component $u_{1,3}$.

In the second case, we have $I = \{1\}$ and $J = \{1, 2\}$. We simulate $p_{1,2}$ as before and we assign a uniform and random value to c_1 , thanks to the presence of the random bit $r_{1,3}$.

In the third case, since c_1 depends on the random bit $r_{1,3}$ which does not appear in the computation of c_2 , and c_2 depends on the random bit $r_{2,3}$ which does not appear in the computation of c_1 , we can simulate both shares as random and independent values.

This reasonings are at the base of the security proof that we provide in the chapter, showing that, after a clever placement of the registers, the ISW multiplication scheme is $t - \text{SNI}$ against glitches.

Furthermore, our theoretical contributions are completed by practical evaluations, taken from [Fau+18].

Related work

Numerous other works in the literature tried to achieve higher-order side-channel security in the presence of physical default, and, in particular, of glitches.

Reparaz *et al.* with their work on Consolidated Masking Scheme [Rep+15], studied links between the ISW multiplication scheme and the concept of TIs. The authors proposed an approach to mask implementations in hardware using only $t + 1$, where t indicates the security order. Such a scheme was later instantiated in [Cnu+16] to implement a masked AES.

The Domain Oriented Masking (DOM), introduced by Gross *et al.* in [GMK16; GMK17], has as well the goal of allowing secure masking in hardware with only $t + 1$ shares. In this case, the authors propose two masked multipliers, one for independent inputs and one for dependent ones. In particular, the first one is closely related to the ISW multiplication scheme, while the second one is more efficient in terms of randomness cost.

With the Unified Masking Approach (UMA) in [GM17], Gross and Mangard follow the concept of DOM, proposing a new hardware multiplication scheme which provides the most efficient randomness use so far. More precisely, according to the different values of d , their scheme extends the parallel masking algorithm from [Bar+17] with the randomness optimizations proposed in [Bel+16] and the DOM multiplier.

We underline that none of the works just mentioned provide security proofs at arbitrary orders. Additionally, the authors in [Moo+19], exhibited composability flaws for each of the works aforementioned, showing how the lack of formal proofs makes flaws more difficult to detect.

The implementation of Section 5.4.2 is the only published algorithm that is jointly robust against glitches and composable at arbitrary orders with $n + 1$ shares. We note

that the parallel masking algorithm introduced by Barthe *et al.* in [Bar+17] exploits the same “compute partial products - refresh - compress” structure as our implementation in Figure 5.1. So despite more specialized to software implementations, it can lead to similar 2-cycle hardware implementations.

After the publication of our study in [Fau+18], many other works followed it up.

Meyer *et al.* in [DBR19] described an information theoretic condition, called-glitch immunity, which is both necessary and sufficient for ensuring security in the presence of glitches. Additionally, in order to unify security concepts, they revised the notions of (Strong) Non-Interference in an information theoretic manner as well.

Dhooghe *et al.* in [DNR19] studied in further detail TIs in the robust-probing model. They proved that TI provides first-order and higher-order univariate security in the glitch-robust probing model. In particular, the authors revised the properties of TIs in terms of composability. They showed a link between uniformity and the notion of Non-Interference and relaxed the notion of non-completeness, in order to facilitate the design of secure expansion and compression functions.

Lastly, Cassiers *et al.* in [Cas+20] extended the simulatability framework of Belaïd *et al.* [Bel+16] and provided a framework of probing security with a compositional security notion against glitches. In the new setting, called Hardware Private Circuits, the authors presented masked gadgets that allow trivial composition with glitches at arbitrary orders.

Other follow up works focused on automatic verification of hardware masked implementations. For instance, in [Blo+18] the authors proposed an automated analysis of masked implementations for detecting the presence of glitches. Their approach consists in estimating the Fourier coefficients of each gate, in order to identify statistical dependence on their inputs. Moreover, the authors in [Bar+18a] provided an alternative method for proving security of masked implementations in the threshold probing model with glitches, by improving their popular tool `maskVerif`.

5.1. A composability notion for Threshold Implementation

In this section, we analyze the case of Threshold Implementations’ (TIs), a type of masking scheme, recalled in Section 2.7, aimed at counteracting power (or electromagnetic) analysis attacks in the presence of glitches.

5.1.1. Pseudo—NI and pseudo—SNI security

We start our analysis by considering an attractive property of Threshold Implementations, i.e., the fact that it benefits not only for security guarantees in the presence of glitches but also for its low randomness requirements. As shown in [Pos+11], some block ciphers can be protected in the TI model with a minimum randomness cost, as the block

size. This interesting property hints that such implementations profit from some sort of composability guarantees, which we aim at investigating in this section.

We begin by considering a function at the core of many efficient S-box decompositions for TIs, i.e., the function $f(x, y, z) = (x \cdot y) \oplus z$, which is commonly called Toffoli gate. The following equation represents a Threshold Implementation at 1st-order of such f with only 3 shares:

$$\begin{aligned} c_1 &= (x_2 \cdot y_2) \oplus (x_2 \cdot y_3) \oplus (x_3 \cdot y_2) \oplus z_2, \\ c_2 &= (x_3 \cdot y_3) \oplus (x_3 \cdot y_1) \oplus (x_1 \cdot y_3) \oplus z_3, \\ c_3 &= (x_1 \cdot y_1) \oplus (x_1 \cdot y_2) \oplus (x_2 \cdot y_1) \oplus z_1. \end{aligned} \tag{5.1}$$

At first sight, we can notice that the addition of the third variable z to the non-linear part $x \cdot y$ ensures the uniformity of the output shares. Such a gadget is 1-probing secure since the output shares c_i 's are independent of x, y and z . However, in a single clock cycle (i.e., intermediate computations such as $x_2 \cdot y_2$, $x_2 \cdot y_3$, \dots do not leak and no probes are allowed on them), it is interesting to remark that it is not 1-SNI, nor 1-NI. As an example, let us suppose to have a single probe on an output share, say on c_1 . Since its computation requires two shares of x and two shares of y , and there is no internal randomness in the gadget, the simulation requires two shares of each input, in contrast to Definition 2.5.

In order to intuitively connect TIs and composable masking schemes, we propose the following variation of existing NI and SNI definitions.

Definition 5.1 (Pseudo-randomized gadgets). *The pseudo-randomization g' of a gadget g is the gadget g modified such that any input share coming from a uniform encoding and appearing only once and as a monomial of degree one in the algebraic circuit description of the gadget g is removed from the gadget inputs and replaced by internal uniform randomness in g' . We denote these monomials as pseudo-randomized monomials.*

Definition 5.2 (Pseudo- t -NI/SNI). *A gadget g is said pseudo- t -NI (respectively, pseudo- t -SNI) if and only if the pseudo-randomization g' of this gadget g is t -NI (resp., t -SNI).*

The new definitions have mainly explanatory purpose at the end of underlining the important conceptual differences between TIs and standard composable masking schemes. Clearly, pseudo-SNI is a weaker notion than SNI and it does not ensure the same composability properties. Its composability guarantees are limited to the case the pseudo-randomized monomials are used carefully, which is indeed what happens in state-of-the-art (1st-order) TIs.

We remark that the gadget of Equation 5.1 is pseudo-1-SNI, since the outputs c_i 's can be simulated thanks to uniform randomness. More formally, we will prove in Section 5.4.1 that this gadget is in reality pseudo-2-SNI.

$(x, y, z) \backslash (c_1, c_2, c_3)$	000	011	101	110	001	010	100	111
000	16	16	16	16	0	0	0	0
010	16	16	16	16	0	0	0	0
100	16	16	16	16	0	0	0	0
111	16	16	16	16	0	0	0	0
001	0	0	0	0	16	16	16	16
011	0	0	0	0	16	16	16	16
101	0	0	0	0	16	16	16	16
110	0	0	0	0	16	16	16	16

Table 5.1.: Number of times that the output shares (c_1, c_2, c_3) occur for a given input (x, y, z) in a 3-share Toffoli TI gate (as given in [Bill15], Section 3.3, Figure 3.1)

For the moment, we show why the gadget of Equation 5.1 fulfills the uniformity property by considering Table 5.1, built according to the techniques used in [Bill15], i.e., by calculating the number of times that the output shares (c_1, c_2, c_3) occur for a given input (x, y, z) . The uniformity property holds if each non-zero entry of the table equals $\frac{2^{n_{in} \cdot (n-1)}}{2^{n_{out} \cdot (n-1)}}$ with $n_{in} = 3$ and $n_{out} = 1$ the function's input and output bit-sizes, respectively, and n the number of shares.

Using the same technique just described, let us now analyze the case of two functions $f_1(x, y, z) = ((x \cdot y) \oplus z, x, y)$ and $f_2(x, y, z) = ((x \cdot y) \oplus z, x, z)$, built using f . By computing similar tables to Table 5.1 for f_1 and f_2 , we can see that the non-zero entries equal 1 (as required by the uniformity property) for f_1 , and 2 for f_2 . This means that the first function's output shares can directly serve as a uniform input sharing for another function, and being safely composed. On the other hand, the second function's output shares are not uniform, and cannot be safely composed. Intuitively, such a result should be expected by observing that f_2 forwards the pseudo-randomized monomial z , which should be used once according to Definition 5.2.

5.1.2. The tradeoff between the number of shares and the cycle count

We complete our investigation on TIs by examining the natural tradeoff between the number of shares and the cycle count. Typically, in order to obtain function decompositions that satisfy the properties of non-completeness and uniformity, TIs mask the inputs with more than $t + 1$ shares, which instead are commonly used in Boolean masking. As an example, in the Toffoli gate presented above, the 1st-order security is obtained

with three shares rather than two. We start our analysis with two main remarks, which allow us to show an implementation of the Toffoli gate in only two shares. First, we notice that the TIs of complex circuits, e.g., S-boxes, are generally constructed by the composition of simpler stages of gadgets. Such stages are divided by memory elements, which prevent the propagation of glitches. Our idea is trying to split the gadgets in more cycles than needed for glitch-freeness. Using this technique, we can, for instance, implement the previous Toffoli gate in two cycles, rather than one. Secondly, we notice that the intermediate stages of the computation do not need to satisfy the correctness property as well. However, it is sufficient that the final result is correct.

In the light of these observations, one possible alternative to the implementation of $f(x, y, z) = (x \cdot y) \oplus z$ is the following one:

$$\begin{aligned} c_1 &= \left[[(x_1 \cdot y_1) \oplus z_1] \oplus (x_1 \cdot y_2) \right], \\ c_2 &= \left[[(x_2 \cdot y_2) \oplus z_2] \oplus (x_2 \cdot y_1) \right], \end{aligned} \tag{5.2}$$

where the $[\cdot]$ parentheses are used to denote the clock cycles.

In the new implementation, the correctness property is satisfied only by the result in the second stage, while the intermediate stage is not correct. However, each stage of this decomposition is non-complete and uniform, as required by TI properties. As in the previous example, this is given by the remark that each stage of the decomposition is pseudo-1-SNI, and the pseudo-randomized monomials are only used once in the circuit.

The implementation of Equation 5.2 is depicted in Figure 5.2, where the circled boxes indicates functions and the rectangles memory elements. We remark how the gate resembles the ISW multiplication [ISW03], where the XOR with z replaces the randomness and makes the gadget pseudo-composable, rather than composable.

5.2. The robust probing model

In the previous section, we saw as first-order TIs are valid means for building first-order glitch-resistant circuits. Unfortunately, the method does not easily scale to higher security orders. Specifically, even though higher-order TIs preserve security against glitches [Bil+14], they do not keep composability properties as well [Rep15].

Let us, for instance, take the following gadget, with inputs x and y , and r_1, r_2, r_3 random elements summing up to 0:

$$\begin{aligned} c_1 &= (x_2 \cdot y_2) \oplus (x_2 \cdot y_3) \oplus (x_3 \cdot y_2) \oplus r_1, \\ c_2 &= (x_3 \cdot y_3) \oplus (x_3 \cdot y_1) \oplus (x_1 \cdot y_3) \oplus r_2, \\ c_3 &= (x_1 \cdot y_1) \oplus (x_1 \cdot y_2) \oplus (x_2 \cdot y_1) \oplus r_3. \end{aligned} \tag{5.3}$$

The gadget represents a variant of Equation 5.1, with the difference that random elements substitute the shares of z . If we do not consider the possibility of physical defaults, the

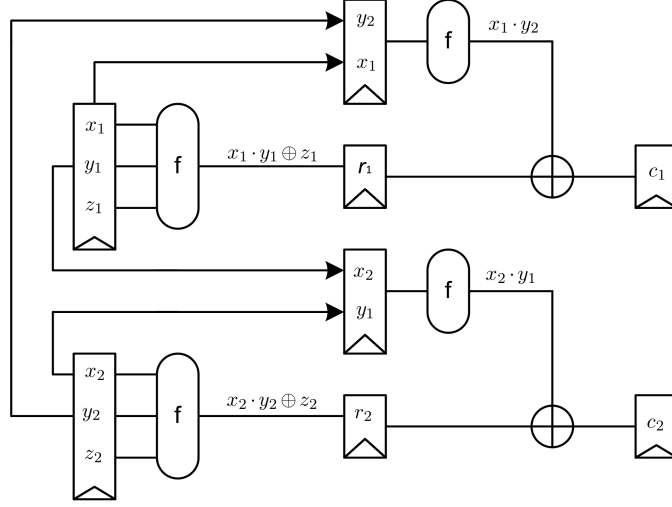


Figure 5.2.: Example of 2-share/2-cycle decomposition of a Toffoli gate [Fau+18]

gadget design implies that the adversary cannot get any information about the internal values $x_i \cdot y_j$ and their intermediate sums. In this setting, this implementation is 2 – SNI. It is not our scope at this moment focusing on the proof, but the arguments of the proof in Section 5.4.1, Proposition 5.5 for the gadget of Equation 5.1 are essentially identical.

If we consider the gadget in a concrete hardware implementation, the assumptions made above are unrealistic, as the computation can leak about intermediate values via glitches or other physical defaults. In this case, the gadget is still probing secure in the presence of glitches thanks to the non-completeness property. However, it is not SNI, since the intermediate randomness can be leaked due to glitches, preventing any simulation according to Definition 2.5.

This example shows that the properties of non-completeness and uniformity are alone not sufficient to ensure composability in the presence of glitches, and it confirms that, with such tools, it is not straightforward to obtain higher-order masked implementations in hardware. The authors in [Rep+15] already pointed out that the addition of refreshing gadgets is essential for guaranteeing composability.

Such a difficulty is not surprising, considering the discussion in the previous section on first-order TIs and pseudo-composability. Univariate attacks in first-order TIs, i.e., attacks with a single probe or targeting a single point in time of the leakage traces, can be prevented by the uniformity property. On the other hand, such a property does not capture multivariate attacks, exploiting multiple probes or targeting multiple points in time of the leakage traces, which are the threat of higher-order security.

At this point, our goal is to provide new tools that can ensure both composability and protection from physical defaults. We could take two main directions. The first one is to exploit the potentials of TIs in preserving security against shares' recombinations and try

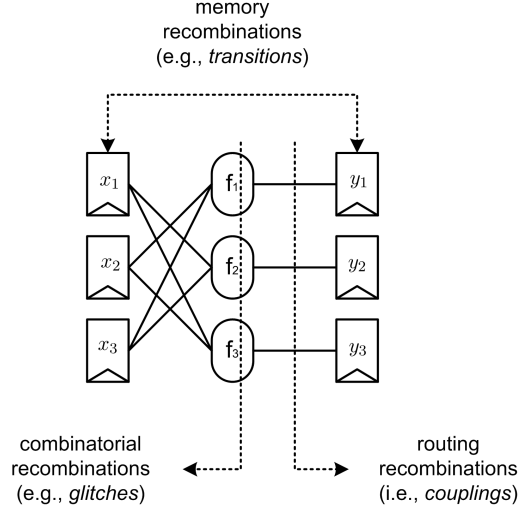


Figure 5.3.: Physical defaults in an example of TI [Bar+17]

to generalize the notion of uniformity in order to scale at high-order. The second option is to base our alternative on the probing model, since the notion of **SN1** already provides a way to argue about composability, and try to generalize the adversarial model such that it covers a wide class of physical defaults. The first option would imply requiring a more global and computationally hard condition to the implementations with the increase of t . We chose to focus on the second one, and in the following, we show that there is a natural generalization of the probing model meeting our security needs.

Modeling physical defaults

In order to describe our new model, we first need to specify how the adversary's power changes when considering hardware implementations. We define in the following three principal physical defaults, which we include in the classical probing model.

We use the example of TI in Figure 5.3 in order to visually represent such physical defaults. First, combinatorial recombinations (e.g., glitches) possibly mix, and consequently recombine, the inputs of the component functions f_i . Second, memory recombinations (e.g., transitions) possibly mix, and consequently recombine, the content of the memory elements in consecutive invocations/cycles. Following the example in Figure 5.3, this can happen if the same memory gate is employed to store the y_i 's by erasing the x_i 's. Third, routing recombinations (i.e., couplings) possibly mix, and consequently recombine, the shares manipulated by adjacent wires.

Our core idea for the generalization of the probing model is to represent adversarial observations as specifically or generically ϵ -extended probes. Generic extensions indicate that the model is independent of the circuit topology, while specific extensions depend

on it. We use such extended probes to define three specific models corresponding to each physical default.

Specific model for glitches. *For any ϵ -input circuit gadget g , combinatorial recombinations (aka glitches) can be modeled with specifically ϵ -extended probes so that probing any output of the gadget allows the adversary to observe all its ϵ inputs.*

The recombination described above is a worst-case scenario. Nevertheless, such a situation occurs often in standard CMOS circuits, as pointed out in [FG05] and [BM16].

As a natural consequence on the topology of robust masked implementation, we need to require to *limit the shares fan-in of each gadget*, i.e., the number of shares of a sensitive variable at its inputs. Consequently, we additionally need that *memory elements separate any composition of gadgets with limited shares fan-in*. Indeed, without the last condition, we would risk that composing gadgets may further increase the shares fan-in. For the reasons above, in the rest of the paper the two requirements just mentioned will be regularly applied to masked circuits topologies.

We underline that the presence of memory elements between composed gadgets makes the probes of some sensitive values looking different through the circuit. In particular, the signal before storage in a register is glitchy and described by extended probes, while the signal stored in a register is stable and represented by standard probes. For instance, this is what happens in output shares, which are typically stored in registers. In this case, the probes on the stable output are counted as output probes, while the ones on their glitchy counterpart are considered internal probes.

Specific model for transitions. *For a memory cell m , memory recombinations (aka transitions) can be modeled with specifically 2-extended probes so that probing m allows the adversary to observe any pair of values stored in 2 of its consecutive invocations.*

This model precisely corresponds to the transition-based leakages given in [Bal+14].

Specific model for couplings. *For any set of adjacent wires $\mathcal{W} = (w_1, \dots, w_d)$, routing recombinations (aka couplings) can be modeled with specifically γ -extended probes so that probing one wire w_i allows the adversary to observe γ wires adjacent to w_i .*

Note that when the parameter γ is set to 0, we are in a situation where no couplings happen. This physical default is the most challenging to deal with, as it is not easy evaluating in practice the parameter γ .

As a last remark, we underline that the models presented above are not assumed to reflect the reality of physical defaults perfectly. Our main goal is instead to capture such defaults in a sufficiently valuable way, in order to help algorithmic designs to achieve better robustness against them. As an example of how the models above translate in

terms of extended-probes on a masked implementation, we can consider a circuit with maximum shares fan-in f . According to the model for glitches, for any standard probe, we have f probes; in the case of transitions, any probe translates into two probes; finally, generic couplings translate any probe into $\gamma + 1$ probes (independently of whether they observe adjacent wires of the circuit).

In the light of the observations above, we say that a gadget is secure in the (λ, τ, γ) -robust t -probing model if:

- the probes are extended with glitches (iff $\lambda = 1$),
- the probes are extended with transitions (iff $\tau = 1$),
- the probes are extended with γ -couplings (for an integer $\gamma \geq 1$),

and we use the same probe extensions in order to define (λ, τ, γ) -robust t -NI/SNI security. The classical t -probing model is thus the $(0, 0, 0)$ -robust t -probing model.

5.3. Physical defaults combination

The physical defaults that we just mentioned can appear simultaneously and combine. For instance, couplings can be combined with transitions if the adversary probes adjacent memory cells. In the same way, couplings can be combined with glitches. Following the example in Figure 5.3, we can imagine that an adversary probes y_1 with a glitch-extended probe, which lets him observe x_2 and x_3 , and that the wire carrying x_2 is coupled with x_1 in Figure 5.3.

Due to the possible combinations just mentioned, the previous model directly implies the following worst-case bound.

Proposition 5.3 (Worst-case generic bound). *Any $2f(\gamma + 1)t$ -probing secure masked circuit with maximum shares fan-in f is $(1, 1, \gamma)$ -robust t -probing secure with generically extended probes.*

The proof of the proposition is trivial, and it merely exploits the fact that any standard probe needs to be “multiplied” by f (because of glitches), by 2 (because of transitions) and by $\gamma + 1$ (because of couplings). This means that one needs $2f(\gamma + 1)t + 1$ shares to obtain robust t -probing secure circuits.

We underline that this proposition only holds for probing security, and not for the stronger properties of NI/SNI. As hinted in Section 5.1.1, and further clarified in the subsequent Section 5.4.2, achieving robust composability requires a more careful analysis of the extended probes’ positions. Moreover, we point out that, by exploiting an appropriate circuit topology, we can reach better results.

Coming back to our example, we can observe for $f = 2$ a loss by a factor $2(\gamma + 1)$, as in Proposition 5.3. Let us try to understand if a factor 2, due the combination of transitions and glitches, should appear as well in the leakage factor.

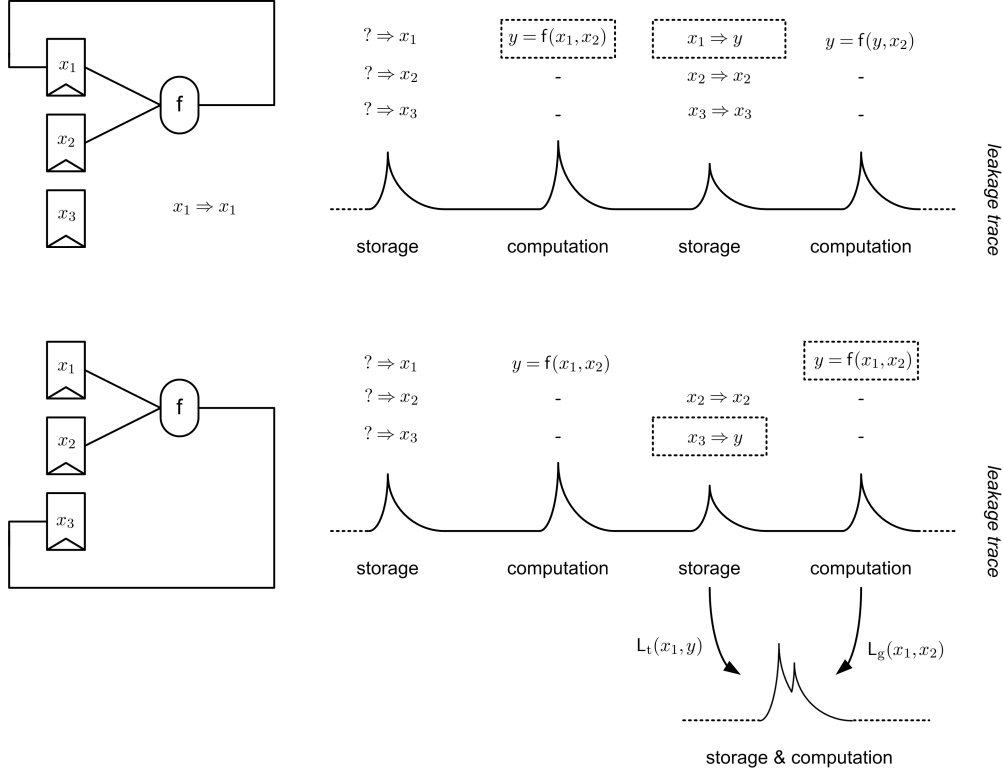


Figure 5.4.: Exemplary combinations of transitions and glitches [Fau+18]

In order to understand better the combinations of transitions and glitches, we consider the circuit examples given in Figure 5.4, where $L_t(\cdot)$ denotes transition-based leakages and $L_g(\cdot)$ glitch-based leakages.

We first notice that glitches can simulate certain transitions. As an example, we consider the upper circuit of the figure, where in the second storage cycle, a transition $x_1 \Rightarrow y$ happens in the top memory cell. Still, a glitch-extended probe on y , permitting an adversary to observe x_1 and x_2 , can simulate this transition, as $y = f(x_1, x_2)$. Therefore, in this case, transitions and glitches do not combine. However, if we consider the lower circuit in the figure, this positive result does not hold anymore, since the $x_3 \Rightarrow y$ transition cannot be simulated by a glitch-extended probe.

As a second remark, we point out that transitions and glitches cannot be combined when the leakage samples corresponding to the storage and computation in a circuit are independent of each other. Such independent leakage samples would correspond to the oversimplified model of the top figure. If that model were perfect, the adversary would have the option to have a glitch-extension or a transition-extension of his probes (giving a factor $2(\gamma + 1)$ rather than $4(\gamma + 1)$ in Proposition 5.3).

In the light of these two observations, we can state that the combination of transitions and glitches in masked implementations is strongly linked to their dependency. Therefore, we can conclude that, in practice, computations within gadgets happen very quickly after the storage, leading these two steps to overlap, as at the bottom of Figure 5.4. Moreover, since the leakage samples given by the combinatorial gates are combined with those of the memory gates, such an overlap can be viewed as a type of parallel implementation, which are usually hard to capture with the probing model and are better considered by the bounded moment model [Bar+17].

At this point, we further notice that the algebraic degree of the combination function between transition-based leakages and glitch-based leakages influences the reduction of the security order in the bounded moment model differently. The authors in [Bar+17], Lemma 1, already showed that a linear combination of $L_t(\cdot)$ and $L_g(\cdot)$ (e.g., a sum in \mathbb{R}) does not decrease this security order. However, a non-linear combination does. In our context, such a non-linear combination of $L_t(\cdot)$ and $L_g(\cdot)$ exactly corresponds to the couplings of Section 5.2. Specifically, couplings typically combine the leakage of adjacent wires (or combinatorial gadgets, memory gates) non-linearly. This fact is shown by the extension factor γ in the probing model and is captured by an algebraic degree $\gamma + 1$ for the combination function in the bounded moment model.

The discussion above leads to the following conjecture, which can guide cryptographic hardware designers.

Conjecture 5.4 (informal). *Any $\max(2, f)t$ -probing secure masked circuit with maximum shares fan-in f is t^{th} -order secure in the bounded moment model if it has transitions \mathcal{E} and glitches but no non-linear combinations of transitions \mathcal{E} and glitches (i.e., couplings).*

In other words, the conjecture hints that if couplings can be kept negligible within an implementation (which depends on the noise level: see [DDF14], Section 4.2), then combinations of glitches and transitions should not affect its concrete security level. In the following, we show experiments that confirm that, in some context, this assumption holds.

5.3.1. Experimental validation

We report here the implementation of a first-order TI of the PRESENT S-box performed in [Fau+18]. It uses two stages, similarly to Figure 5.3, and follows the guidelines in [MW15], Figure 3, in a Xilinx Spartan-6 FPGA, measured on the SAKURA-G board.⁴

The authors tweaked the design in two different ways. First, in place of using six distinct registers $x_1, x_2, x_3, y_1, y_2, y_3$ to store the input and output shares, the same register is used to store x_1 and y_1 . Indeed such a change should lead to first-order leakages due to transitions. Second, the authors refreshed the output of f_1 with uniform randomness

⁴ <http://satoh.cs.uec.ac.jp/SAKURA/index.html>

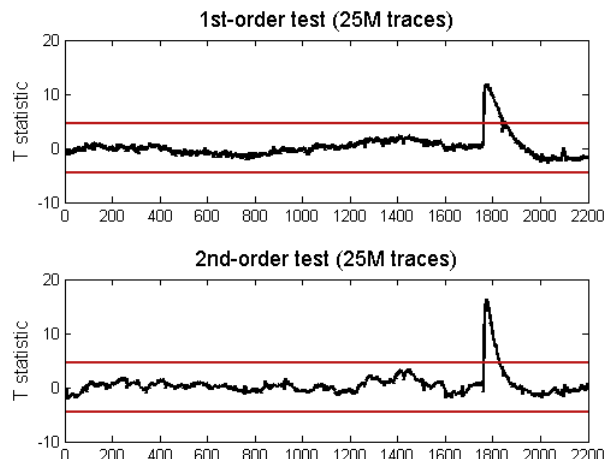


Figure 5.5.: Non-specific t-test results for a first-order TI of the PRESENT S-box tweaked such that the register storing x_1 and y_1 in Figure 5.3 is reused without refreshing [Fau+18]

before storing it in the reused register storing x_1 . This refreshing is not supposed to increase the security order if glitches and transitions combine, since a glitch-extended probe on y_1 provides such additional randomness to the adversary. However, the refreshing should improve the security level in case glitches and transitions are not combined, since the adversary can have a glitch-extended probe on y_1 before this is stored in the register, and a transition-extended probe on y_1 after it is stored in the register.

The authors launched CRI’s non-specific t-test on such implementations to detect differences between the traces corresponding to fixed and random inputs [Coo+13; Goo+11]. The results of these experiments are depicted in Figures 5.5 and 5.6. Figures 5.5 shows the presence of leakage, presumably due to transitions, which is canceled when the refreshing is activated, as shown by Figure 5.6.

This confirms that certain types of transitions do not combine harmfully with glitches, as stated before in this section.

5.4. Concrete constructions

Finally, we now examine a couple of popular constructions from the literature and discuss their security properties in the robust-probing model and how they perform in relation to the previous worst-case predictions.

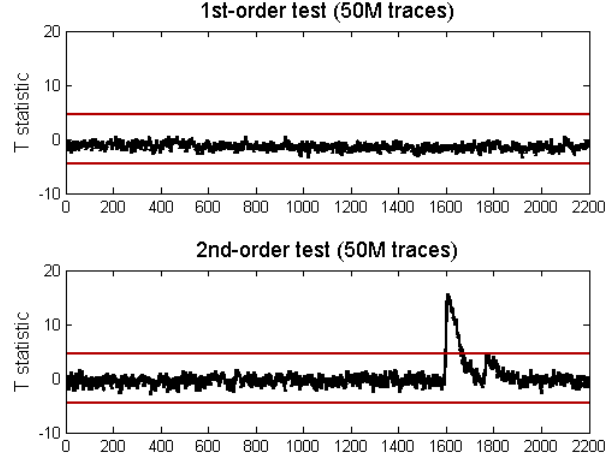


Figure 5.6.: Non-specific t-test results for a first-order TI of the PRESENT S-box tweaked so that the register storing x_1 and y_1 in Figure 5.3 is reused with refreshing [Fau+18]

5.4.1. The Toffoli gate leads to pseudo- $(1,0,0)$ -robust 1-probing security

We start by analyzing the Toffoli gate construction in Section 5.1.1, which is a 1st-order TI gadget typically used in many TI of block cipher S-boxes. We design a hardware implementation example, selecting the registers in order to avoid transition issues so that $\tau = 0$ in the robust probing model. Concretely, the Toffoli gadget is computed in one cycle, and its outputs are stored in memory gates. We first show with Proposition 5.5 that, if we ideally exclude the presence of glitches, the scheme is pseudo-2-SNI.

Proposition 5.5. *The TI implementation in 1 cycle of Equation 5.1 is pseudo-2-SNI.*

Proof. According to Definition 5.2, in order to prove that the gadget in Equation 5.1 is pseudo-2-SNI, we need to prove that its pseudo-randomization, called g' , is 2-SNI. The algorithm g' corresponds to Equation 5.1, with the difference that the inputs are only the shares $x_1, x_2, x_3, y_1, y_2, y_3$ and the values z_1, z_2, z_3 are assigned uniformly at random. Let $\Omega = \{w_1, w_2\}$ be a set of 2 adversarial observations on the pseudo-randomized gadget g' . Since the implementation of the scheme is only in one cycle, there is no intermediate computation and therefore the probes can only lie in one of the following two groups:

- (1) the input shares x_i and y_j with $i, j \in \{1, 2, 3\}$;
- (2) the output shares c_1, c_2, c_3 .

Let t_1 (resp., t_2) be the number of observations on the input (resp., output) values (with $t_1 + t_2 \leq 2$). We first define two sets of indices I and J such that $|I| \leq t_1$ and $|J| \leq t_2$.

and the values of the probes can be perfectly simulated given only the knowledge of $(x_i)_{i \in I}$ and $(y_j)_{j \in J}$. The sets are constructed as follows:

- Initially I and J are empty.
- For every probe as in group (1), add i to I and j to J .

Since the adversary is allowed to make at most t_1 probes on the input values, it holds that $|I| \leq t_1$ and $|J| \leq t_1$. In order to prove the SNI property, we next show the simulation phase, by distinguishing the three different cases listed below.

1. If $t_1 = 2$, then the probes w_1 and w_2 are both in group (1) and, by definition of the set I , the simulator has access to the observed shares x_i and y_i .
2. If $t_1 = 1$ and $t_2 = 1$, then w.l.o.g. w_1 is in group (1) and w_2 is in group (2). By definition of the sets I and J , the simulator has access to the observed shares, therefore w_1 can be perfectly simulated. As for w_2 , thanks to the random value z_h with $h \in \{1, 2, 3\}$, the probe can be simulated by assigning a random and independent value.
3. Finally, if $t_2 = 2$, then w_1 and w_2 are both in group (2) and they are of the form $x_i y_i + x_i y_j + x_j y_i + z_i$ with $i \in \{1, 2, 3\}$. Since the (pseudo-randomized) shares of z appearing in their computation are different in each output share, the simulator can also assign w_1 and w_2 to a random and independent value.

In all the cases listed above, the probes w_1 and w_2 can be perfectly simulated with t_1 shares of the input. We finally note that if $|\Omega| = 1$, then the simulation of the probe trivially follows the procedure of one of the previous cases. Therefore we conclude that the gadget \mathbf{g}' is 2-SNI, completing the proof. \square

By combining this result and Proposition 5.3, it directly follows that this TI gadget is pseudo-(1,0,0)-robust 1-probing secure with 3 shares. In other words, it uses an additional share to prevent glitches, precisely as we anticipated in the worst-case analysis. Once again we underline that the gadget is not pseudo-(1,0,0)-robust 1-SNI, since glitch-extended probes cannot be simulated with one share per input. However, for first order of security, when the uniformity condition in Section 5.1.1 is fulfilled, it is sufficient to have the security of TIs for arguing that a full cipher is pseudo-2-SNI without glitches. By using Proposition 5.3, we achieve that they are also (1,0,0)-robust 1-probing secure.

5.4.2. The ISW is (1,0,0)–robust $t - \text{SNI}$ with $t + 1$ shares in 2 cycles

We now move our analysis to the ISW multiplication algorithm, which has been the first probing secure multiplication algorithm, introduced in the seminal work of Ishai *et al.* [ISW03]. The algorithm was proved to be (0,0,0)-robust $t - \text{SNI}$ in [Bar+16] using $t + 1$ shares. The authors in [RP10] generalizes the original algorithm to larger fields, given that the refreshing is adjusted to make it composable, as discussed in [Cor+13]. In the following, we use a slight variation of the algorithm, which is represented in Algorithm 5.1.

Algorithm 5.1 Modified ISW multiplication algorithm with $n \geq 2$ shares

Input: shares $(a_i)_{1 \leq i \leq n}$ and $(b_i)_{1 \leq i \leq n}$, such that $\bigoplus_i a_i = a$ and $\bigoplus_i b_i = b$.

Output: shares $(c_i)_{1 \leq i \leq n}$, such that $\bigoplus_i c_i = a \cdot b$.

```

1: for  $i = 1$  to  $n$  do
2:   for  $j = i + 1$  to  $n$  do
3:      $r_{i,j} \xleftarrow{\$} \mathbb{F}_{2^n}$ ;
4:      $u_{i,j} \leftarrow r_{i,j} \oplus a_i \cdot b_j$ ;
5:      $u_{j,i} \leftarrow r_{i,j} \oplus a_j \cdot b_i$ ;
6: for  $i = 1$  to  $n$  do
7:    $c_i \leftarrow a_i \cdot b_i \oplus \bigoplus_{j=1, j \neq i}^n u_{i,j}$ ;

```

Essentially, we organize the intermediate results slightly differently than originally, in order to have a scheme better suited to prevent physical defaults.

We show in the following that, when moving to higher-orders, the ISW multiplication beats the worst-case bound, and consequently provides an excellent solution for robust and composable gadgets. In particular, we formally prove that the scheme is secure also in the presence of glitches, i.e., it is $(1,0,0)$ -robust $t - \text{SNI}$, if we precisely follow the guidelines of Section 5.2 and limits the shares fan-in to 1.

To this end, we consider an implementation of the ISW multiplication in two cycles, depicted in Figure 5.1 for the case with 3 shares and security order 2. A description of the algorithm at any order can be obtained by using the notations of Section 5.1.2 and adding one level of brackets around the $u_{j,i}$ and $u_{i,j}$ variables of Algorithm 5.1, indicating the operations performed in the first cycle, and a second level of brackets around the c_i variables, indicating the operations performed in the second cycle.

Our proof uses the specific model for glitches, that exploits this particular circuit topology. Concretely, it means that for the implementation illustrated in Figure 5.1, the adversary can access the three types of probes listed in the following:

- Internal (3-extended) probes $p_{i,j}$ on the $u_{i,j}$'s giving access to three shares, i.e., a_i, b_j and the corresponding value of the randomness matrix.
- Internal (3-extended) probes p_i on the c_i 's giving access to $u_{i,1}, u_{i,2}, u_{i,3}$.
- Output (non-extended) probes on the c_i 's giving only access to one share.

Note that in this model, if an adversary wants to obtain a single internal value (e.g., an $r_{i,j}$) he will need to use an extended probe including this value. Note also that, despite giving 3-extended probes to the adversary, we do not break the shares fan-in limit of 1. Moreover, the c_i shares appear twice in the list: either as internal probes, which can be glitch-extended, or as external probes, which are not glitchy since stored in an additional memory element. We underline that, despite not being necessary for probing security, the additional output memory elements storing the c_i shares are strictly necessary in

order to ensure composability. In the following, we formalize the security proof.

Proposition 5.6. *The multiplication gadget in Algorithm 5.1 implemented in two cycles (one for the $u_{i,j}$ values, a second one for the c_i values) is $(1,0,0)$ -robust t -SNI.*

Proof. Let $\Omega = (\mathcal{I}, \mathcal{O})$ be a set of t adversary's observations respectively on the internal and on the output values, where $|\mathcal{I}| = t_1$ and in particular $t_1 + |\mathcal{O}| \leq q$. We construct a perfect simulator of the adversary's probes, which makes use of at most t_1 shares of the secrets x and y .

Let w_1, \dots, w_q be the probed values. According to the specific model for glitches presented in Section 5.2, the possible internal extended probes can be classified in the following groups:

- (1) $p_{i,j} := (a_i, b_j, r_{i,j})$ with $i, j = 1, \dots, t+1$
- (2) $p_i := (u_{i,1}, \dots, u_{i,t+1})$ with $i = 1, \dots, t+1$.

On the other hand, since the output shares are stored in registers, glitches do not affect them and so the possible probes on the output shares are, as in the non-robust probing model, the c_i with $i = 1, \dots, t+1$, as in Algorithm 5.1.

We define two sets of indices I and J such that $|I| \leq t_1$, $|J| \leq t_1$ and the values of the probes can be perfectly simulated given only the knowledge of $(x_i)_{i \in I}$ and $(y_i)_{i \in J}$. The sets are constructed as follows.

- Initially I and J are empty.
- For every probe as in group (1) add i to I and j to J .
- For every probe as in group (2) add i to I and moreover for every probe of the form $p_{j,i}$ add j to J .

Since the adversary is allowed to make at most t_1 internal probes, it holds that $|I| \leq t_1$ and $|J| \leq t_1$.

We now show the simulation phase. First of all, the simulator assigns a random value to every $r_{i,j}$ entering in the computation of any probe. Then we consider an observed value w_h in group (1). In this case, by definition of I and J the simulator has access to a_i and b_j and we distinguish three cases:

- If $i = j$, the simulator assigns $r_{i,i}$ to 0 and then perfectly simulates w_h using a_i and b_i .
- If $j \in I$ and $i \in J$, then by definition the adversary has probed also $p_{j,i}$ or p_i and p_j . Therefore, in any case, the adversary has already probed a value containing in its computation the random bit $r_{i,j}$. The simulator then perfectly simulates w_h using a_i , b_j and the $r_{i,j}$ assigned previously.
- In all the other cases, $r_{i,j}$ does not enter in the computation of any other probe, and therefore the simulator can assign w_h to a random and independent value.

As for a probe w_h in group (2), by definition $i \in I, J$. So the simulator can perfectly

compute the i th-component of the probe using a_i, b_i . For each of the remaining j th-components of p_i we distinguish the following cases.

- If $j \in J$ and $j \notin I$, then the adversary has already probed $p_{i,j}$, which can be simulated as in the first phase and entirely used as j th-component of w_h .
- If $j \in J$ and $j \in I$, then the adversary has already probed $p_{i,j}$ or p_j or $p_{j,i}$. In the first case the simulator follows the previous step. In both the latter cases, $r_{i,j}$ was assigned in the preliminary phase and can be used with a_i and b_j to simulate the j th-component of w_h .
- If $j \notin J$, the simulator assigns to the j th-component of p_i a random and independent value: indeed, the bit $r_{i,j}$ involved in the computation of such a component is not used in any other probe.

We conclude the proof by showing how to simulate a probe w_h in the output values. We notice that since in this case the probes are as in the traditional probing model, the proof is really similar to the one of Proposition 2 in [Bar+15b]. We have to take into account the following two cases:

- If the attacker has observed also some of the internal values, then the partial sums previously probed are already simulated. As for the remaining terms, we note that, by definition of the scheme, there always exists one random bit $r_{k,l}$ in w_h , which does not appear in the computation of any other observed element. Therefore the simulator can assign to w_h a random and independent value.
- If the attacker has only observed output shares, then we point out that by definition each of them is composed by t random bits and at most one of them can enter in the computation of each other output variable c_i . Since the adversary may have previously probed at most $t - 1$ of them, there exist one random bit $r_{k,l}$ in w_h , which does not appear in the computation of any other observed value. Thus the simulator can assign to w_h a random and independent element, completing the proof.

□

We insist that, despite our 2-cycle implementation is directly inspired by the ISW construction, the previous proofs of ISW-like multiplications do not imply the new proof. Indeed, the extended probes provide additional information to the adversary per time, compared to the software setting analyzed by [RP10] and follow-up works.

We underline that the simulation for the output values profits the careful distribution of the random bits in the different registers. In particular, each output share depends on a number of distinct random bits equal to the security order, and these random bits appear a second time in the computation of only one different output share each. This fact allows us to simulate the output probes with a random and independent value, and therefore to use the required number of input shares in order to satisfy the definition of SNI. The main overhead of this glitch-robust and composable multiplication is the need

of $n^2 + n$ registers, to store the partial products in a first cycle and compress the output in a second cycle.

It is an interesting open problem to determine whether robust and composable gadgets could be obtained in two cycles with fewer registers and/or randomness than in this section, e.g., by arranging the operations differently, or if such optimizations can only be obtained by increasing the number of cycles.

5.4.3. Glitch locality principle

As we pointed out in the discussion above, robust and composable implementations of the ISW multiplication require to store their outputs c_i 's in memory gates, in order to stop the propagation of glitches in the circuit. This remark leads to the following formalization.

Proposition 5.7. *If a gadget g storing its outputs in registers is $(1,0,0)$ -robust t - NI and t - SNI (without glitches), then it is also $(1,0,0)$ -robust t - SNI.*

Proof. By separating the probes between t_1 internal and t_2 output ones, we have that: (i) the internal probes can be simulated with t_1 shares per input since the gadget is $(1,0,0)$ -robust t_1 -probing secure (with $t_1 \leq q$), and (ii) the t_2 probes can be simulated with t_1 input shares since the gadget is t - SNI without glitches. \square

This proposition shows that if registers are inserted after each $(1,0,0)$ -robust t - NI gadget, the problem of glitches represents an issue only at the level of the internal computation. In this case, a designer can deal with composability separately. However, we underline that glitches and composability are not independent issues since glitch-robust t -probing security is not enough for proving the lemma. As shown in [Moo+19], some form of simulatability, captured by the glitch-robust NI notion, is needed.

5.5. Practical security evaluation

We report here the practical security evaluations performed in [Fau+18]. The authors implemented the 2-cycle architecture of Figure 5.1 in a Xilinx Spartan-6 FPGA for $n = 2$ and 3 shares, using the same setup as in Section 5.3.1. Based on such a setup, and since the only interest is the security order of our designs, the authors launched CRI's non-specific t-test to recognize differences between the traces corresponding to fixed inputs and random inputs [Coo+13; Goo+11]. When $n = 2$, the experiments show second-order leakages with 1 million measurements, as reported in Figure 5.7. When $n = 3$, 10 millions measurements are performed and it is exploited the tweak proposed in [Sta17], Section 3.2, which consists in reproducing 50 times the measurement of 250,000 traces and averaged them in order to mitigate the noise amplification due to masking, and to speed up the detection. This permitted to detect third-order leakages, as reported

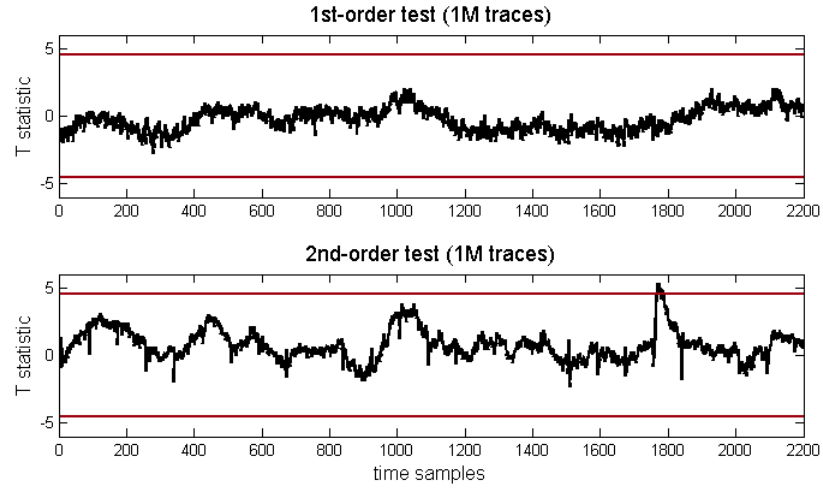


Figure 5.7.: Non-specific t-test results of PRESENT S-box for $n = 2$ shares [Fau+18]

in Figure 5.8. Therefore, since our implementations are composable, we can conclude that a combination of such higher-order hardware gadgets will still be robust against glitches and maintain their security for complete, e.g., block cipher, implementations, as confirmed experimentally for the cipher SIMON in [Roy+15].

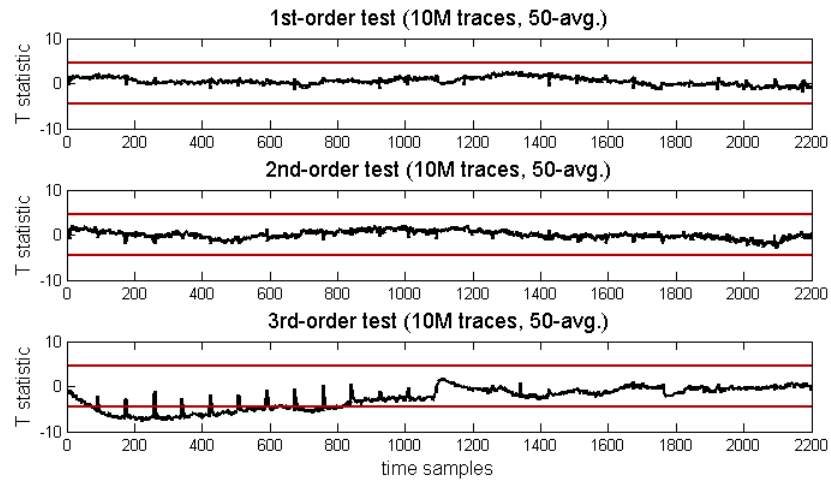


Figure 5.8.: Tweaked non-specific t-test results of PRESENT S-box for $n = 3$ shares [Fau+18]

5.6. Conclusions

While software-oriented masked implementations already benefit from many leakage models and the use of formal proofs for analyzing their security, such tools are not yet well established for hardware-oriented masked implementations. The lack of a suitable model that formally treats the local security and composability of masked gadgets in the presence of physical defaults is one of the causes of such a difference. Consequently, many security claims of proposed masked circuits in hardware have been, before this work, mainly supported by engineering considerations and informal reasoning of probing security in the presence of glitches.

The robust probing model that we introduced in this chapter has the goal of filling this gap. Not only it provides formal guidelines to implementation designers when dealing with physical defaults. Additionally, it presents arguments in order to produce combined robustness against physical defaults and composability guarantees. We underline that robust and composable gadgets are sufficient for designing higher-order secure masked circuits. However, it is not always required that all gadgets in an implementation satisfy both properties [Bel+16]. In this context, our results are open to optimization, by exploiting, for instance, formal methods to analyze full algorithms [Bar+15b]

Our new model makes it also possible to analyze and demonstrate security guarantees of masked implementations in a versatile way, by differently choosing the parameters λ, τ and γ . By tuning such parameters, indeed, an implementer can decide the level of security he wants to achieve, reducing implementation surprises and performance overheads.

The extensive analysis conducted in [Moo+19] on hardware-oriented masking schemes shows that many gadgets in the literature, which are not proven secure in the robust probing model, present flaws of security at a local and compositional level for arbitrary orders, confirming the need of our model.

Lastly, we point out that the introduction of the robust probing model is also helpful when one cannot leverage the local security order analysis of single gadgets, and instead has to deal directly with the complexity of full implementations. Indeed, the model facilitates the analysis of physical defaults in masking schemes with automated tools, as discussed in [Blo+18]. Moreover, its incorporation in the other existing formal tools, such as [Bar+15b], to analyze large implementations, is another interesting research direction.

6. Masking lattice-based cryptographic primitives

With this chapter, we complete our studies with a focus on the protection of algorithms employed in lattice-based constructions, which are considered in the field of post-quantum cryptography.

Motivation

The field of post-quantum cryptography is increasingly growing in response to the rising threat of quantum computers, which would make most contemporary cryptography obsolete. The NIST post-quantum standardization process has reached its second round [ST17], and explicitly mentioned the simplicity of ensuring side-channel security guarantees of the schemes as one evaluation metric [NIS16]. Lattice-based cryptography represents the largest share among the submissions thanks to its advantages, such as the reasonable parameter sizes, the simple implementations, and the strong security guarantees given by advanced constructions, like identity-based encryption and homomorphic encryption.

In this context, studying the security of the practical implementation of lattice-based cryptosystems is becoming an increasingly interesting research direction, however still rather understudied. Indeed, in contrast to standard symmetric and asymmetric cryptography, where the application of masking schemes has been widely examined and achieves reasonable levels of security and efficiency, most lattice-based cryptosystems received less attention in these regards.

In the field of lattice-based cryptography, the Learning With Errors (LWE) problem is at the core of numerous encryption and key exchange schemes. Many of such LWE-based schemes require sampling from a discrete Gaussian distribution. Several works propose methods for generating and implementing discrete Gaussian samplers. For small standard deviations, the binomial sampler introduced in [Alk+] and implemented in [AJS16] is the best option. Its design allows constant run time and easy implementation, and it does not require to precompute tables. On the other hand, the side-channel protection of such a sampler is still subject of on-going research, and, in particular, only few publications address the resistance against power analysis attacks. An example is the work in [Ode+18], where the authors provide a binomial sampler protected against a first-order adversary. However, the scheme provided is not easily extendable to higher security orders, and, in

the case study of [Ode+18], it represents the bottleneck of the implementation, causing a performance overhead of more than 400%.

Such reduced performances derive mainly from the use of different masking schemes. A binomial sampler commonly works by taking the difference between the Hamming weight of two uniformly random bit vectors. Such initial uniform randomness is usually generated by a pseudo random number generator (PRNG). Since many PRNGs rely on Boolean operations, in the initial stage it is convenient to adopt the Boolean masking. However, the subsequent operations in the lattice scheme are mostly linear arithmetic operations, for which the Arithmetic masking is more suited. In order to connect these two steps, a crucial algorithm of a protected sampler is a Boolean-to-arithmetic (B2A) masking conversion. Previous works, as [Ode+18] and [Bar+18a], recognized the development of such a conversion algorithm as a significant challenge for ensuring efficient protection of lattice-based cryptography.

In the light of the observations above, our primary goal in this chapter is to study the integration of physical countermeasures to a binomial sampler for an arbitrary security order. To reach this scope efficiently, we aim at providing new conversion algorithms from Boolean to Arithmetic masking, which, for the parameters adopted in lattice-based cryptography, outperform previous algorithms.

Contribution

In this chapter, we propose two masked binomial samplers for lattice-based encryption and key exchange schemes, protected for arbitrary orders. Both samplers support arbitrary moduli, therefore they can be used by a wide set of NIST submissions, e.g., NEWHOPE [Alk+], LIMA [Sma+17], Saber [D'A+17], Kyber [Ava+17], HILA5 [Saa17], or Ding Key Exchange [Din+17]. The first sampler is a higher-order extension of the method from [Ode+18]. The second one is a variant that uses bit-slicing, which increases the throughput additionally.

We give an intuitive explanation of the contribution, by quickly describing the bit-sliced sampler. The algorithm takes in inputs two κ -bit vectors, x and y , which are masked pseudo-random bits, output of a Boolean-masked SHAKE. Then, the algorithm performs the difference between the Hamming weights of such vectors, and finally a conversion algorithm B2A_q transforms the result into an Arithmetic masked output modulo a prime q .

More precisely, the sampler is structured as in Figure 6.1. First, the routine **HW** computes the Hamming weight of x using bit-slicing. Successively, **SubHW** directly subtracts from the result the Hamming weight of y and **ConstAdd** adds the constant κ to correctly map the sign of the difference. After the B2A_q conversion, κ is subtracted to recover correctly the result.

Most existing B2A conversion algorithms consider a modulo power of two, i.e., of the form 2^k . However, lattice-based schemes often require the modulo to be a prime

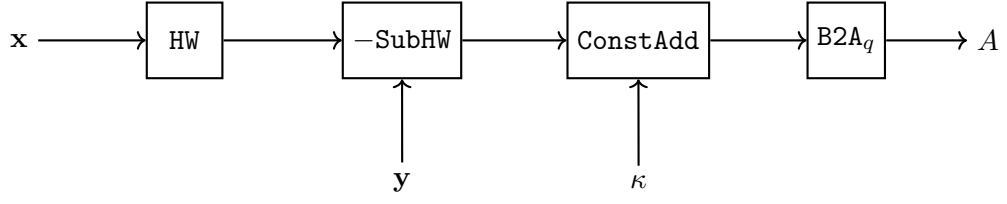


Figure 6.1.: Structure of the new bit-sliced sampler

which can be arbitrarily chosen. For this reason, we also investigate the conversion algorithms for prime moduli. First, we revise the quadratic B2A algorithm of [CGV14], by exploiting some of the ideas defined in [Bar+18a]. This leads to A2B_q and B2A_q algorithms providing the best asymptotic run time complexity to date, i.e., $\mathcal{O}(n^2 \log k)$, where k denotes the bit size of the operands and n the number of shares. Second, we introduce a new B2A_q conversion scheme with run time complexity $\mathcal{O}(n^2 \cdot k)$. While its asymptotic complexity is worse than our quadratic adaptation of [Bar+18a], it still significantly decreases the conversion time for relevant values of k . Moreover, for certain parameters (e.g., $n \geq 11$ for $k = 32$), the new algorithm outperforms the common B2A algorithms for power-of-two moduli.

Our new B2A_q conversion works with a recursive approach. We give in the following an example for security order $t = 2$ and shares $n = 3$. First, the subroutine B2A_{q-Bit}⁽²⁾ takes as inputs the first two shares x_1, x_2 of the input x and produces the shares Sh₁, Sh₂ defined as follows:

$$\begin{aligned} \text{Sh}_1 &= (x_1 - \text{rnd}_1) - 2((x_1 - \text{rnd}_1) \cdot x_2) + x_2 \\ \text{Sh}_2 &= \text{rnd}_1 - 2(\text{rnd}_1 \cdot x_2) \end{aligned}$$

where $\text{rnd}_1 \xleftarrow{\$} \mathbb{F}_q$. Such shares are injected in the subroutine B2A_{q-Bit}⁽³⁾, along with the third input share x_3 , which outputs the shares A'_1, A'_2, A'_3 calculated as follows

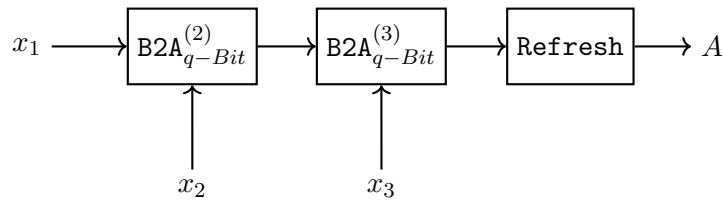


Figure 6.2.: Structure of the new B2A_q, for q prime and $t = 2$

$$\begin{aligned} A'_1 &= (\text{Sh}_1 - \text{rnd}_2) - 2(\text{Sh}_1 - \text{rnd}_2) \cdot x_3 + x_3 \\ A'_2 &= (\text{Sh}_2 - \text{rnd}_2) - 2(\text{Sh}_2 - \text{rnd}_2) \cdot x_3 \\ A'_3 &= \text{rnd}_2 - 2 \cdot \text{rnd}_2 \cdot x_3 \end{aligned}$$

where $\text{rnd}_2 \xleftarrow{\$} \mathbb{F}_q$. It is easy to see the correctness of the approach. The sum of the shares, after canceling out the randomness, gives the following result:

$$\begin{aligned} A'_1 + A'_2 + A'_3 &= x_1 \oplus x_2 - 2 \cdot x_1 x_3 \oplus 2 \cdot x_2 x_3 \\ &= x_1 \oplus x_2 \oplus x_3 = x \end{aligned}$$

Since $\text{B2A}_{q\text{-}Bit}^{(i)}$, for $i \in \mathbb{N}$, is a $t - \text{NI}$ gadget, in order to obtain an overall $t - \text{SNI}$ conversion, the outputs (A'_1, A'_2, A'_3) need to be further processed by a $t - \text{SNI}$ refreshing scheme **Refresh**, which produces the final output shares of the conversion algorithm (A_1, A_2, A_3) .

In the chapter, we provide formal proofs that all our proposed algorithms are $t - \text{SNI}$, with $n = t + 1$ shares.

The implementation of our constructions on an ARM Cortex-M4F micro-controller given in [Sch+19] shows that our new B2A_q conversion increases the performance of the samplers respect to the adaptation of [Bar+18a] up to a factor of 46. Additionally, our new bit-sliced sampler improves the performance over a generalized version of the approach of [Ode+18] up to a factor of 15. The combination of both methods results in the currently most efficient masked binomial sampler.

Related Work

Barthe *et al.* in [Bar+18a] proposed, for the first time, a provably-secure uniform sampler for arbitrary orders. However, compared to binomial samplers, their sampler requires different implementation difficulties. Therefore such a result is challenging to use in the protection of a binomial sampler. In the same year, Oder *et al.* in [Ode+18] developed a protected implementation of NEWHOPE [Alk+], which is one of the submissions to the NIST post-quantum standardization process. In their construction, the authors make use of a masked binomial sampler highly optimized for first-order probing security. Interestingly, their results show that the protection of the binomial sampler can heavily influence the performance of the whole scheme, leaving as an open problem the investigation of a more efficient masked binomial sampler.

As we already underlined, to achieve this goal, the conversion algorithms from Boolean to Arithmetic masking assume a crucial role. Many works in the literature study this type of conversion schemes [BCZ18; Bir+17; CGV14; Cor+15; Cor17; Deb12; HT19; KRJ14;

SMG15; WH17]. However, these schemes focus uniquely on power-of-two moduli, since they mostly target symmetric cryptosystems, and they are, therefore, not convenient to adopt in lattice-based schemes.

Oder *et al.* in [Ode+18] gave some basic solutions for **A2B** and **B2A** conversions modulo a generic q . However, the authors did not rigorously formalize the algorithms, and the security of their approach is restricted to only a first-order adversary. The first conversion algorithm for arbitrary moduli and high-order probing security was introduced in [Bar+18a], and it is based on the cubic conversion of [CGV14].

The authors introduced a new Boolean-masked addition algorithm that works modulo a prime q with $2^k \geq 2q$. According to this scheme, they modified both the **A2B** and **B2A** algorithms from [CGV14] to deal with moduli different than a power-of-two. The security proof that the authors provide refers only to an adaptation of the **A2B** algorithm from [CGV14] with cubic complexity.

Notation

In the rest of the chapter, q represents a prime number, k the bit size of the conversion, κ the bit size of the input vectors of the samplers, and, as in the rest of this thesis, n is the number of shares and t the security order. Conventionally, we denote with **B2A** the standard conversion from Boolean to arithmetic masking and with **B2A_q** the same transformation for prime moduli. Moreover, Boolean encoding is represented in the lower case, while the upper one is adopted for Arithmetic encoding. Operations on the whole encoding, i.e., vector of shares, are denoted in bold and performed share-wise, e.g., $\mathbf{z} = \mathbf{x} \oplus \mathbf{y}$. We indicate the k bits of a share as $(x_i^{(k)} \dots x_i^{(1)}) = x_i$, i.e., the least-significant bit (LSB) of x_i is written as $x_i^{(1)}$ (resp. $\mathbf{x}^{(1)}$ denotes the LSB of \mathbf{x}).

6.1. New Conversion Algorithms from Boolean to Arithmetic masking

In the following sections we first present our techniques to improve the previous conversion algorithms from Boolean to Arithmetic masking, which are used in the masked implementation of binomial samplers.

6.1.1. Improved higher-order **B2A_q** from [Bar+18b]

We start by analyzing how it is possible to adjust the **B2A_q** conversion algorithm with quadratic complexity from [CGV14] to work with prime moduli. The idea that we discuss in this section was already proposed in [Bar+18b] and it is represented in Algorithm 6.1, but it lacked a formal security analysis.

First of all, we recall how the conversion in [CGV14] works. The algorithm starts by initializing the shares $(A_i)_{1 \leq i \leq n-1}$ with random samples in \mathbb{F}_{2^k} . Such shares are then used to produce a random encoding \mathbf{A}' , with shares $\sum_{i=1}^n A'_i = -\sum_{i=1}^{n-1} A_i \pmod{2^k}$. Then, using an Arithmetic-to-Boolean (A2B) conversion algorithm, such an encoding is converted into a Boolean one, with shares $\bigoplus_i y_i = \sum_i A'_i \pmod{2^k}$. The shares y_i are finally summed to the input encoding \mathbf{x} resulting in:

$$\bigoplus_i z_i = \bigoplus_i x_i + \bigoplus_i y_i = x - \sum_{i=1}^{n-1} A_i \pmod{2^k}.$$

At this point, the function $\text{FullXOR}: \mathbb{F}_{2^k}^n \mapsto \mathbb{F}_{2^k}$ [CGV14] securely decodes z and the last share of \mathbf{A} is set to $A_n = \text{FullXOR}(\mathbf{z})$. The correctness of the approach follows quickly:

$$\sum_{i=1}^n A_i = \sum_{i=1}^{n-1} A_i + (x - \sum_{i=1}^{n-1} A_i) = x \pmod{2^k}.$$

In the same paper [CGV14] it is presented an improved version of the aforementioned algorithm as well. Its core idea is to use recursion. The n input shares A_i are divided into two halves of $\lfloor \frac{n}{2} \rfloor$ and $\lceil \frac{n}{2} \rceil$. Then the algorithm recursively calls itself for the two halves and the resulting encodings are then added together as

$$\begin{aligned} x &= (A_1 + \dots + A_{\lfloor n/2 \rfloor}) + (A_{\lfloor n/2 \rfloor + 1} + \dots + A_n) \\ &= (y_1 \oplus \dots \oplus y_{\lfloor n/2 \rfloor}) + (z_1 \oplus \dots \oplus z_{\lceil n/2 \rceil}) \\ x &= x_1 \oplus \dots \oplus x_n. \end{aligned} \tag{6.1}$$

We now give the algorithmic representation of the modified conversion, and we prove its $t - \text{SNI}$ property. This improvement facilitates a fair comparison with our new algorithms. Algorithm 6.2 represents the original scheme from [Bar+18b]. We refer to Appendix A for the routines used in the algorithms. The shares are added sequentially using SecArithBoolModp , leading to run-time complexity $\mathcal{O}(n^3 \cdot \log k)$.

Algorithm 6.1 SecBoolArithModp [Bar+18b]

Input: $\mathbf{x} = (x_i)_{1 \leq i \leq n} \in \mathbb{F}_2^k$ such that $\bigoplus_i x_i = x \in \mathbb{F}_2^k$
Output: $\mathbf{A} = (A_i)_{1 \leq i \leq n} \in \mathbb{F}_q$ such that $\sum_i A_i = x \pmod{q}$

- 1: $(A_i)_{1 \leq i \leq n-1} \xleftarrow{\$} \mathbb{F}_q$
- 2: $(A_i)'_{1 \leq i \leq n-1} \leftarrow q - (A_i)_{1 \leq i \leq n-1}$
- 3: $A'_n \leftarrow 0$
- 4: $\mathbf{y} \leftarrow \text{SecArithBoolModp}(\mathbf{A}')$
- 5: $\mathbf{z} \leftarrow \text{SecAddModp}(\mathbf{x}, \mathbf{y})$
- 6: $A_n \leftarrow \text{FullXOR}(\mathbf{z})$

Algorithm 6.2 SecArithBoolModp (cubic) [Bar+18b]

Input: $\mathbf{A} = (A_i)_{1 \leq i \leq n} \in \mathbb{F}_q$ such that $\sum_i A_i = x \bmod q \in \mathbb{F}_q$

Output: $\mathbf{x} = (x_i)_{1 \leq i \leq n} \in \mathbb{F}_2^k$ with $2^k > 2q$ such that $\bigoplus_i x_i = x$

- 1: $(x_i)_{1 \leq i \leq n} \leftarrow 0$
 - 2: **for** $j = 1$ to n **do**
 - 3: $(y_i)_{1 \leq i \leq n} \leftarrow (A_j, 0, \dots, 0)$
 - 4: $\mathbf{y} \leftarrow \text{RefreshXOR}(\mathbf{y}, k)$
 - 5: $\mathbf{x} \leftarrow \text{SecAddModp}(\mathbf{x}, \mathbf{y}, k)$
-

Algorithm 6.3 SecArithBoolModp (quadratic)

Input: $\mathbf{A} = (A_i)_{1 \leq i \leq n} \in \mathbb{F}_q$ such that $\sum_i A_i = x \bmod q \in \mathbb{F}_q$

Output: $\mathbf{x} = (x_i)_{1 \leq i \leq n} \in \mathbb{F}_2^k$ with $2^k > 2q$ such that $\bigoplus_i x_i = x$

- 1: **if** $n=1$ **then**
 - 2: $x_1 \leftarrow A_1$
 - 3: $(y_i)_{1 \leq i \leq \lfloor n/2 \rfloor} \leftarrow \text{SecArithBoolModp}((A_i)_{1 \leq i \leq \lfloor n/2 \rfloor})$
 - 4: $(y_i)_{1 \leq i \leq n} \leftarrow \text{RefreshXOR}((y_1, \dots, y_{\lfloor n/2 \rfloor}, 0, \dots, 0), k)$
 - 5: $(z_i)_{1 \leq i \leq \lceil n/2 \rceil} \leftarrow \text{SecArithBoolModp}((A_i)_{\lfloor n/2 \rfloor + 1 \leq i \leq n})$
 - 6: $(z_i)_{1 \leq i \leq n} \leftarrow \text{RefreshXOR}((z_1, \dots, z_{\lceil n/2 \rceil}, 0, \dots, 0), k)$
 - 7: $\mathbf{x} \leftarrow \text{SecAddModp}(\mathbf{y}, \mathbf{z})$
-

To improve this, we make use of the recursive construction explained above in Equation (6.1). More precisely, we split the input encoding of **SecArithBoolModp** into two sets of $\lfloor n/2 \rfloor$ and $\lceil n/2 \rceil$ elements. For each subset, we execute a new call of **SecArithBoolModp**, and we add up the results. This strategy provides the advantage that each subroutine processes a smaller number of shares, and consequently the complexity of the refresh and of the addition are reduced. The overall complexity of the conversion is $\mathcal{O}(n^2 \cdot \log k)$.

To derive an analogous quadratic B2A_q conversion, we replace the call of the routine **SecArithBoolModp** with the A2B_q algorithm with quadratic complexity, given in Algorithm 6.3. The $t - \text{SNI}$ refresh **RefreshXOR** from [Bar+18b] is used in place of the **Expand** function from [CGV14] in order to map the $\lfloor n/2 \rfloor$ (resp. $\lceil n/2 \rceil$) Boolean shares to the n shares needed for the secure masked addition. We first pad with zeros the Boolean encodings and then we refresh the resulting n shares. An exemplary structure for the case $n = 4$ is depicted in Figure 6.4.

Security analysis

The $t - \text{SNI}$ security of **SecBoolArithModp** in Algorithm 6.1, when using the quadratic version of **SecArithBoolModp**, relies on the fact that **SecArithBoolModp** itself is $t - \text{SNI}$.

Before proceeding with proving that Algorithm 6.3 is t -SNI, we give the following lemma.

Lemma 6.1. *Given a circuit \mathcal{C} as in Figure 6.3, where \mathbf{f}, \mathbf{g} are t -SNI gadgets and \mathbf{h} is t -NI, the circuit \mathcal{C} is t -SNI.*

Proof. Let $\Omega = (\mathcal{I}, \mathcal{O})$ be the set of adversarial observations on \mathcal{C} , where \mathcal{I} are the probes on the internal values and \mathcal{O} the ones on the output values, with $|\mathcal{I}| + |\mathcal{O}| \leq t$. In particular, let $\mathcal{I}_{\mathbf{f}}$ be the set of probes on \mathbf{f} , $\mathcal{I}_{\mathbf{g}}$ the set of probes on \mathbf{g} and $\mathcal{I}_{\mathbf{h}}$ be the set of probes on \mathbf{h} , with $|\mathcal{I}_{\mathbf{f}} \cup \mathcal{I}_{\mathbf{g}} \cup \mathcal{I}_{\mathbf{h}}| \leq |\mathcal{I}|$.

We prove the existence of a simulator which can simulate the adversary's view by using at most $|\mathcal{I}|$ input shares, analyzing the circuit from right to left.

Gadget \mathbf{h} : Since \mathbf{h} is t -NI and $|\mathcal{I}_{\mathbf{h}} \cup \mathcal{O}| \leq t$, then there exist two observation sets $\mathcal{S}_1^{\mathbf{h}}, \mathcal{S}_2^{\mathbf{h}}$ such that $|\mathcal{S}_1^{\mathbf{h}}| \leq |\mathcal{I}_{\mathbf{h}} \cup \mathcal{O}|$, $|\mathcal{S}_2^{\mathbf{h}}| \leq |\mathcal{I}_{\mathbf{h}} \cup \mathcal{O}|$ and the gadget can be simulated using at most $|\mathcal{S}_1^{\mathbf{h}}| + |\mathcal{S}_2^{\mathbf{h}}|$ shares of the inputs.

Gadget \mathbf{f} : Since \mathbf{f} is t -SNI and $|\mathcal{I}_{\mathbf{f}} \cup \mathcal{S}_1^{\mathbf{h}}| \leq t$, then there exists an observation set $\mathcal{S}^{\mathbf{f}}$ such that $|\mathcal{S}^{\mathbf{f}}| \leq |\mathcal{I}_{\mathbf{f}}|$ and the gadget can be simulated using at most $|\mathcal{S}^{\mathbf{f}}|$ shares of the inputs.

Gadget \mathbf{g} : Since \mathbf{g} is t -SNI and $|\mathcal{I}_{\mathbf{g}} \cup \mathcal{S}_2^{\mathbf{h}}| \leq t$, then there exists an observation set $\mathcal{S}^{\mathbf{g}}$ such that $|\mathcal{S}^{\mathbf{g}}| \leq |\mathcal{I}_{\mathbf{g}}|$ and the gadget can be simulated using at most $|\mathcal{S}^{\mathbf{g}}|$ shares of the inputs.

Combining the previous steps of the simulation, we can claim that \mathcal{C} can be simulated by using at most $|\mathcal{S}^{\mathbf{f}} \cup \mathcal{S}^{\mathbf{g}}| \leq |\mathcal{I}_{\mathbf{f}}| + |\mathcal{I}_{\mathbf{g}}| \leq |\mathcal{I}|$ shares of the inputs, completing the proof. \square

Proposition 6.2. *SecArithBoolModp in Algorithm 6.3 is t -SNI, for $t = n - 1$.*

Proof. In order to prove SecArithBoolModp to be t -SNI we iteratively split the circuit in sub-circuits \mathcal{C}_i , for $i = 2, \dots, n$, where $\mathcal{C}_n := \text{SecArithBoolModp}$, as in Figure 6.4, and we prove the statement by induction on $i \in \mathbb{N}$.

We remark that \mathcal{C}_2 is of the form of the circuit in Figure 6.3. Indeed RefreshXOR is t -SNI and SecAddModp is t -NI, as proven in [Bar+18b]. Therefore thanks to Lemma 6.1, the circuit \mathcal{C}_2 is t -SNI.

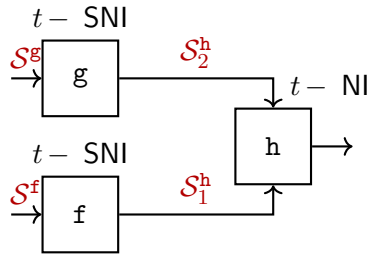
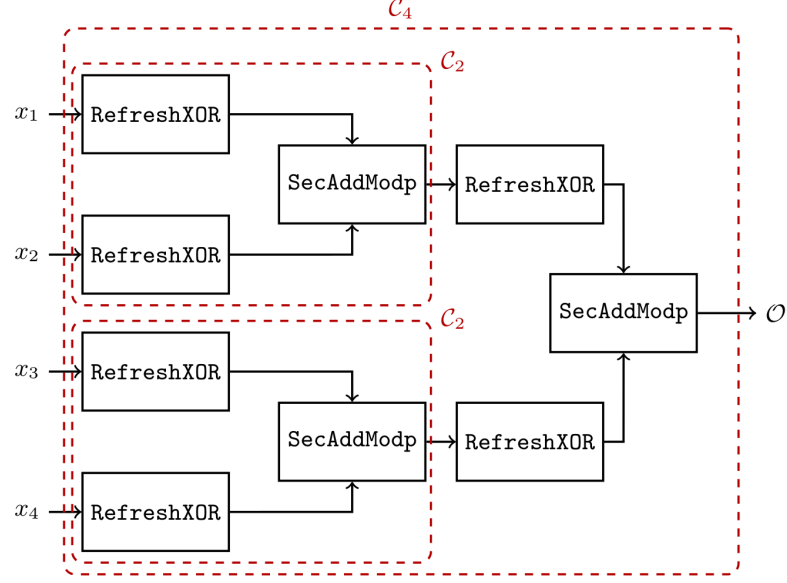


Figure 6.3.: Recurrent scheme in Algorithm 6.3


 Figure 6.4.: Structure of SecArithBoolModp in Algorithm 6.3 for $n = 4$

Let us suppose now that \mathcal{C}_{n-2} is $t - \text{SNI}$ and we prove the statement for \mathcal{C}_n .

Since the composition of $t - \text{SNI}$ gadgets is still $t - \text{SNI}$, as pointed out in [Bar+15a], we know that both $\mathcal{C}_{n-2}((A_i)_{1 \leq i \leq \lfloor n/2 \rfloor})$ and $\mathcal{C}_{n-2}((A_i)_{\lfloor n/2 \rfloor + 1 \leq i \leq n})$ composed with the gadget RefreshXOR are $t - \text{SNI}$. Therefore the circuit \mathcal{C}_n can be represented as the circuit in Figure 6.3, where \mathbf{f} is substituted by $\text{RefreshXOR}(\mathcal{C}_{n-2}((A_i)_{1 \leq i \leq \lfloor n/2 \rfloor}))$, \mathbf{g} by $\text{RefreshXOR}(\mathcal{C}_{n-2}((A_i)_{\lfloor n/2 \rfloor + 1 \leq i \leq n}))$, and \mathbf{h} by SecAddModp . From Lemma 6.1 we conclude therefore that \mathcal{C}_n is $t - \text{SNI}$, completing the proof. \square

6.1.2. A new B2A_q conversion algorithm for $x \in \mathbb{F}_2$

In this section, we present our work on a new B2A_q conversion algorithm for $x \in \mathbb{F}_2$. The core idea of the new scheme is to exploit the following arithmetic property that Boolean-masked bits (x_1, x_2) with $x_1 \oplus x_2 = x$ provide:

$$x = x_1 + x_2 - 2 \cdot x_1 \cdot x_2. \quad (6.2)$$

We use such a relation to transform these Boolean-masked bits to an Arithmetic encoding modulo q .

When the variables are in \mathbb{F}_2 , our goal is to securely transform the Boolean shares $(x_1, x_2) \in \mathbb{F}_2$ with $x_1 \oplus x_2 = x$ into Arithmetic shares (A_1, A_2) with $A_1 + A_2 = x \pmod{q}$, for some arbitrary modulo q . For this purpose, the Boolean shares are transformed to the Arithmetic encodings $B_1 + B_2 = x_1$ and $C_1 + C_2 = x_2$. This results in the

Algorithm 6.4 $\text{SecB2A}_{q-\text{Bit}}$ (simple)

Input: $\mathbf{x} = (x_i)_{1 \leq i \leq n} \in \mathbb{F}_2$ such that $\bigoplus_i x_i = x$
Output: $\mathbf{A} = (A_i)_{1 \leq i \leq n} \in \mathbb{F}_q$ such that $\sum_i A_i = x \pmod{q}$
 1: $(A_i)_{1 \leq i \leq n} \leftarrow 0$
 2: **for** $j = 1$ to n **do**
 3: $(B_i)_{1 \leq i \leq n} \leftarrow (x_j, 0, \dots, 0)$
 4: $\mathbf{B} \leftarrow \text{RefreshADD}(\mathbf{B}, q)$
 5: $\mathbf{C} \leftarrow \text{SecMul}(\mathbf{A}, \mathbf{B})$
 6: $\mathbf{A} \leftarrow \mathbf{A} + \mathbf{B} - 2 \cdot \mathbf{C}$

following relation:

$$\begin{aligned}
 x &= x_1 \oplus x_2 = (B_1 + B_2) \oplus (C_1 + C_2) \\
 &= (B_1 + B_2) + (C_1 + C_2) - 2 \cdot (B_1 + B_2) \cdot (C_1 + C_2).
 \end{aligned}$$

Before deriving the final Arithmetic encoding \mathbf{A} , we have to sample a fresh random element $R \in \mathbb{F}_q$ to secure the shared multiplication $(B_1 + B_2) \cdot (C_1 + C_2)$. Now, it is easily possible to compute the shares as:

$$\begin{aligned}
 A_1 &= B_1 + C_1 + R - 2 \cdot B_1 \cdot C_1 - 2 \cdot B_1 \cdot C_2 \\
 A_2 &= B_2 + C_2 - R - 2 \cdot B_2 \cdot C_1 - 2 \cdot B_2 \cdot C_2.
 \end{aligned}$$

In order to extend this simple first-order method to arbitrary orders, we need to adopt a suited refresh and multiplication algorithm, as represented in Algorithm 6.4.

This straightforward method has the drawback to be really expensive and inefficient for increasing security orders. Indeed the algorithm has a run time complexity of $\mathcal{O}(n^3)$, i.e., cubic in the number of shares. In order to increase the performance of such simple algorithm, we apply the following improvements. First, we notice that using the $t - \text{SNI}$ refresh RefreshADD for every iteration round is not necessary. Instead, it is sufficient to use a $t - \text{NI}$ refresh per round and a single $t - \text{SNI}$ refresh at the end of the algorithm. Such a change decreases the complexity of the refresh for every iteration from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$. As a second improvement, in place of multiplying two complete encodings as $\text{SecMul}(\mathbf{A}, \mathbf{B})$, we do not refresh x_j and instead compute the component-wise multiplication $\mathbf{A} \cdot x_j$. At this point, we underline that this option is possible only thanks to the aforementioned $t - \text{NI}$ refresh, which ensures that the encoding \mathbf{A} is independent of x_j . Such a second change allows saving another operation with $\mathcal{O}(n^2)$ and reducing it to $\mathcal{O}(n)$. Moreover, we can save $n - 1$ operations of the addition $\mathbf{A} + x_j$. Our last improvement consists of varying the number of considered shares in each iteration, e.g., for $j = 2$, the operations are done on two shares.

We give the optimized conversion in Algorithm 6.5, which now has a run time complexity $\mathcal{O}(n^2)$. Its structure is depicted in Figure 6.5.

Algorithm 6.5 $\text{SecB2A}_{q-\text{Bit}}$ (optimized)

Input: $\mathbf{x} = (x_i)_{1 \leq i \leq n} \in \mathbb{F}_2$ such that $\bigoplus_i x_i = x \in \mathbb{F}_2$

Output: $\mathbf{A} = (A_i)_{1 \leq i \leq n} \in \mathbb{F}_q$ such that $\sum_i A_i = x \pmod{q}$

$\mathbf{A} \leftarrow \text{B2A}_{q-\text{Bit}}(\mathbf{x})$

$\mathbf{A} \leftarrow \text{RefreshADD}(\mathbf{A}, q)$

Algorithm 6.6 $\text{B2A}_{q-\text{Bit}}$

Input: $\mathbf{x} = (x_i)_{1 \leq i \leq n} \in \mathbb{F}_2$ such that $\bigoplus_i x_i = x \in \mathbb{F}_2$

Output: $\mathbf{A} = (A_i)_{1 \leq i \leq n} \in \mathbb{F}_q$ such that $\sum_i A_i = x \pmod{q}$

1: $\mathbf{A} \leftarrow x_1$

2: **for** $j = 2$ to n **do**

3: $\mathbf{A} \leftarrow \text{B2A}_{q-\text{Bit}}^{(j)}(\mathbf{A}, x_j)$

Correctness

We prove the correctness of $\text{B2A}_{q-\text{Bit}}$, since the refreshing afterwards does not change the decoded output. The proof is based on the following property, already mentioned in Equation (6.2). Given $x_1, x_2 \in \mathbb{F}_2$, the XOR between the two bits can be written as $x_1 \oplus x_2 = x_1 + x_2 - 2 \cdot x_1 \cdot x_2$. Generalizing to the case of n values, it is easy to see that

$$\bigoplus_{i=1}^n x_i = (((x_1 \oplus x_2) \oplus x_3) \dots) \oplus x_n = \bigoplus_{i=1}^{n-1} x_i + x_n - 2 \cdot \bigoplus_{i=1}^{n-1} x_i \cdot x_n. \quad (6.3)$$

Now, adding the output shares of Algorithm 6.6 we get

$$\begin{aligned} \sum_{i=1}^n A_i &= \sum_{i=1}^n B_i - 2 \cdot \sum_{i=1}^n B_i \cdot x_n + x_n = \sum_{i=1}^{n-1} A_i - 2 \cdot \sum_{i=1}^{n-1} A_i \cdot x_n + x_n \\ &= \bigoplus_{i=1}^{n-1} x_i - 2 \cdot \bigoplus_{i=1}^{n-1} x_i \cdot x_n + x_n. \end{aligned}$$

which, for Equation 6.3, is exactly $\bigoplus_{i=1}^n x_i$.

Security analysis

In the following propositions we show that our conversion scheme $\text{SecB2A}_{q-\text{Bit}}$ in Algorithm 6.5 satisfies the t -SNI property, when $t = n - 1$.

Proposition 6.3. $\text{B2A}_{q-\text{Bit}}^{(2)}$ in Algorithm 6.7 is 1-NI.

Proof. Let us suppose that the adversary has exactly 1 probe w in Algorithm 6.7. This belongs to one of the following possible groups:

Algorithm 6.7 $\text{B2A}_{q\text{-}Bit}^{(n)}$

Input: $\mathbf{A} = (A_i)_{1 \leq i \leq n-1} \in \mathbb{F}_q$ such that $\sum_i A_i = x$; $x_n \in \mathbb{F}_2$

Output: $\mathbf{C} = (C_i)_{1 \leq i \leq n} \in \mathbb{F}_q$ such that $\sum_i C_i = (x \oplus x_n) \bmod q$

- 1: $B_n \xleftarrow{\$} \mathbb{F}_q$
 - 2: $B_1 \leftarrow A_1 - B_n \bmod q$
 - 3: **for** $j = 2$ to $n - 1$ **do**
 - 4: $R \xleftarrow{\$} \mathbb{F}_q$
 - 5: $B_j \leftarrow A_j - R \bmod q$
 - 6: $B_n \leftarrow B_n + R \bmod q$
 - 7: **for** $j = 1$ to n **do**
 - 8: $C_j \leftarrow B_j - 2 \cdot (B_j \cdot x_n) \bmod q$
 - 9: $C_1 \leftarrow C_1 + x_n \bmod q$
-

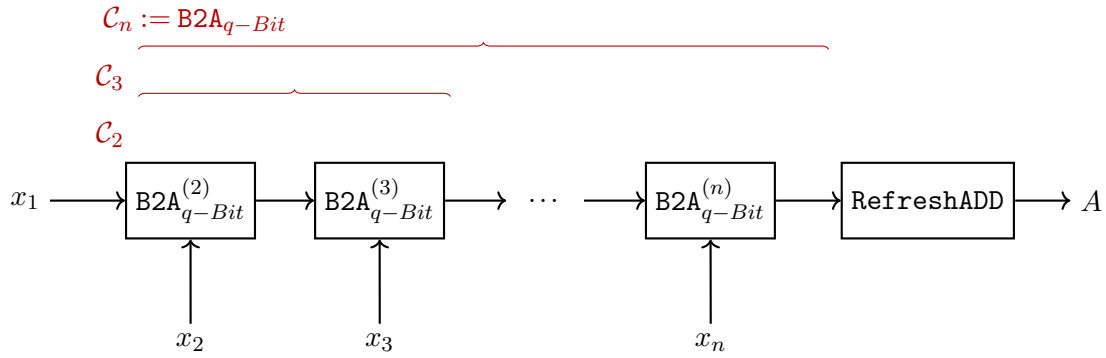


Figure 6.5.: Structure of $\text{SecB2A}_{q\text{-}Bit}$ in Algorithm 6.5

- (1) x_1, x_2
- (2) B_2
- (3) $B_1 := A_1 - B_2 = x_1 - B_2$
- (4) $B_1 \cdot x_2 = (x_1 - B_2) \cdot x_2, b := B_1 - 2(B_1 \cdot x_2)$
- (5) $B_2 \cdot x_2, B_2 - 2(B_2 \cdot x_2)$
- (6) $b + x_2 = (B_1 - 2(B_1 \cdot x_2)) + x_2$

The simulation of the probe w , by using at most 1 share of the inputs, is straightforward.

- If the probe w is in group (1) the values can be simulated as x_1 (resp. x_2) as in the real algorithm.
- If the probe w is in one of the groups from (2) to (6), thanks to the presence in the computation of w of the random values B_1 or B_2 , it is simulated by assigning it to a random and independent value in \mathbb{F}_q .

□

We remark that the algorithm is not t -SNI. Indeed, if an adversary probes the output share $(B_1 - 2(B_1 \cdot x_2)) + x_2 = ((x_1 - B_2) - 2((x_1 - B_2) \cdot x_2)) + x_2$ and the internal value B_2 , then the adversary gets the knowledge of two inputs shares, contradicting the definition of t -SNI.

In the following proposition we prove that the algorithm $\text{B2A}_{q\text{-}Bit}$ is t -NI.

Proposition 6.4. $\text{B2A}_{q\text{-}Bit}$ in Algorithm 6.6 is t -NI.

Proof. In the following, we denote with \mathcal{C}_i the execution of Algorithm 6.6 until the i^{th} iteration of the **for**, for $i = 2, \dots, n$, as indicated in Figure 6.5. Note that $\mathcal{C}_n := \text{B2A}_{q\text{-}Bit}$.

We prove the statement by induction on the value i . From Proposition 6.3 the condition is satisfied for the case of $i = 2$.

Let now assume that \mathcal{C}_i is $(i-1)$ -NI for all $i \leq n-1$ and we show that under this condition \mathcal{C}_n is t -NI as well. First of all, let us denote with $A_j^{(i)}$ the output shares of \mathcal{C}_i for all $i \leq n-1$ and with $j = 1, \dots, n$. We can classify the internal and output values of \mathcal{C}_n in the following groups:

- (1) B_j for $j = 2, \dots, n, r$
- (2) A_1^{n-1}
- (3) $b_1 := A_1^{n-1} - B_2 - \dots - B_j$, with $j = 2, \dots, n$
- (4) $A_j^{(n-1)} - r_j =: b_j$, with $j = 2, \dots, n$
- (5) $A_1^{(n)} = (A_1^{n-1} - B_2 - \dots - B_n) + x_n = b_1 + x_n$
- (6) $A_j^{(n)} = (A_j^{(n-1)} - r_j) - 2((A_j^{(n-1)} - r_j) \cdot x_n) = b_j - 2(b_j \cdot x_n)$
- (7) x_n

Let us suppose w.l.o.g. that an adversary has Ω probes on \mathcal{C}_n with $|\Omega| = t = t_1 + \dots + t_n$, where $t_1 + \dots + t_i$ are the probes on \mathcal{C}_i . We show that the adversary's observation on \mathcal{C}_n can be simulated by using at most t shares of the input.

We first construct a set of indexes I . For each probe in group (5) or group (6), we add n to I . The simulation follows the steps below.

- **Step 1.** The probes in group (1) are simulated by assigning them to a random and independent value in \mathbb{F}_q .
- **Step 2.** Since by hypothesis $\text{B2A}_{q\text{-}Bit}^{(n-1)}$ is t -NI, the probes in group (2) can be simulated by using at most $t_1 + \dots + t_{n-1}$ shares.
- **Step 3.** If a probe is in group (3) and at least one of the B_2, \dots, B_j is not in Ω , then the values can be assigned to a random and independent value. Otherwise, if $B_2, \dots, B_j \in \Omega$, then since $\text{B2A}_{q\text{-}Bit}^{(n-1)}$ is t -NI the probes can be simulated by using at most $t_1 + \dots + t_{n-1}$ shares of the input and the assigned values of B_2, \dots, B_j in Step 1. Otherwise, if a sum $A_1^{n-1} - B_2 - \dots - B_k \in \Omega$, with $k < j$, and $B_{k+1}, \dots, B_j \in \Omega$, then the values can be computed as in the real execution of the algorithm, by using the

values B_{k+1}, \dots, B_j assigned in Step 1 and the simulated sum $A_1^{n-1} - B_2 - \dots - B_k$ in one of the phases of this Step.

- **Step 4.** If a probe is in group (4) and $r_j \notin \Omega$, then the values can be assigned to a random and independent value. Otherwise, if $r_j \in \Omega$, then since $\mathbf{B2A}_{q-\text{Bit}}^{(n-1)}$ is $t - \text{NI}$ the probes can be simulated by using at most $t_1 + \dots + t_{n-1}$ shares and the value r_j assigned in Step 1.
- **Step 5.** If a probe is in group (5) and $b_1 \in \Omega$, then by construction $n \in I$ and we can compute the value as in the algorithm, by using the b_1 simulated at Step 3 and x_n . Otherwise, if $b_1 \notin \Omega$, we can simulate b_1 as in Step 3, by using at most $t_1 + \dots + t_{n+1}$ input shares and x_n , since $n \in I$.
- **Step 6.** If a probe is in group (6) and $b_j \in \Omega$, then by construction $n \in I$ and we can compute the value as in the algorithm, by using the b_j simulated in Step 4 and x_n . Otherwise, if $b_j \notin \Omega$, we can simulate b_j as in Step 4, by using at most $t_1 + \dots + t_{n+1}$ input shares and x_n , since $n \in I$.
- **Step 7.** If a probe is in group (7), by construction $n \in I$ and we can trivially simulate x_n .

In all the steps listed above, we showed that the simulation uses at most t input shares, as required from Definition 2.4, completing the proof. \square

Before proceeding with the next proposition, we remind that the refreshing scheme added at the end of $\mathbf{SecB2A}_{q-\text{Bit}}$ is an algorithm presented in [Bar+18b] and proven to be $t - \text{SNI}$. The $t - \text{SNI}$ security of $\mathbf{SecB2A}_{q-\text{Bit}}$ relies exactly on the introduction of this gadget after the computation of $\mathbf{SecB2A}_{q-\text{Bit}}$. We see below a more detailed security proof.

Proposition 6.5. $\mathbf{SecB2A}_{q-\text{Bit}}$ in Algorithm 6.5 is $t - \text{SNI}$.

Proof. The $t - \text{SNI}$ security of $\mathbf{SecB2A}_{q-\text{Bit}}$ easily follows from the fact that $\mathbf{B2A}_{q-\text{Bit}}$ is $t - \text{NI}$ and $\mathbf{RefreshADD}$ is $t - \text{SNI}$.

Let $\Omega = (I, \mathcal{O})$ be the set of probes on $\mathbf{SecB2A}_{q-\text{Bit}}$, where I_1 are the probes on the internal wires of $\mathbf{B2A}_{q-\text{Bit}}$ and I_2 are the probes on the internal wires of $\mathbf{RefreshADD}$, with $|I| = |I_1| + |I_2| \leq t_1$ and $|I| + |\mathcal{O}| \leq t$.

Since $\mathbf{RefreshADD}$ is $t - \text{SNI}$ and $|I_2 \cup \mathcal{O}| \leq t$, then there exists an observation set \mathcal{S}^2 such that $|\mathcal{S}^2| \leq |I_2|$ and the gadget can be simulated from its input shares corresponding to the indexes in \mathcal{S}^2 .

Since $\mathbf{SecB2A}_{q-\text{Bit}}$ is $t - \text{NI}$ and $|I_1 \cup \mathcal{S}^2| \leq |I_1 \cup I_2| \leq t$, then it exists an observation set \mathcal{S}^1 such that $|\mathcal{S}^1| \leq |I_1 \cup \mathcal{S}^2|$ and the gadget can be simulated from its input shares corresponding to the indexes in \mathcal{S}^1 .

Now, by composing the simulators that we have for the two gadgets $\mathbf{RefreshADD}$ and $\mathbf{SecB2A}_{q-\text{Bit}}$, all the probes of the circuit can be simulated from $|\mathcal{S}^1| \leq |I_1| + |I_2| \leq t_1$

Algorithm 6.8 RefreshADD (based on RefreshXOR [Bar+18b])

Input: $\mathbf{A} = (A_i)_{1 \leq i \leq n} \in \mathbb{F}_q$ such that $\sum_i A_i = x \pmod q$, modulo q

Output: $\mathbf{B} = (B_i)_{1 \leq i \leq n} \in \mathbb{F}_q$ such that $\sum_i B_i = x \pmod q$

```

1:  $\mathbf{B} \leftarrow \mathbf{A}$ 
2: for  $i = 1$  to  $n - 1$  do
3:   for  $j = 1 + i$  to  $n$  do
4:      $R \xleftarrow{\$} \mathbb{F}_q$ 
5:      $B_i \leftarrow B_i + r$ 
6:      $B_j \leftarrow B_j - r$ 
    
```

Algorithm 6.9 SecB2A_q

Input: $\mathbf{x} = (x_i)_{1 \leq i \leq n} \in \mathbb{F}_{2^k}$ such that $\bigoplus_i x_i = x \in \mathbb{F}_{2^k}$

Output: $\mathbf{A} = (A_i)_{1 \leq i \leq n} \in \mathbb{F}_q$ such that $\sum_i A_i = x \pmod q$

```

1:  $\mathbf{A} \leftarrow \text{SecB2A}_{q-\text{Bit}}((\mathbf{x} \gg (k-1)) \wedge 1)$ 
2: for  $j = 2$  to  $k$  do
3:    $\mathbf{B} \leftarrow \text{SecB2A}_{q-\text{Bit}}((\mathbf{x} \gg (k-j)) \wedge 1)$ 
4:    $\mathbf{A} \leftarrow 2 \cdot \mathbf{A} + \mathbf{B} \pmod q$ 
    
```

shares of the input x and therefore, according to Definition 2.5, the circuit in Figure 6.5 is $t - \text{SNL}$.

□

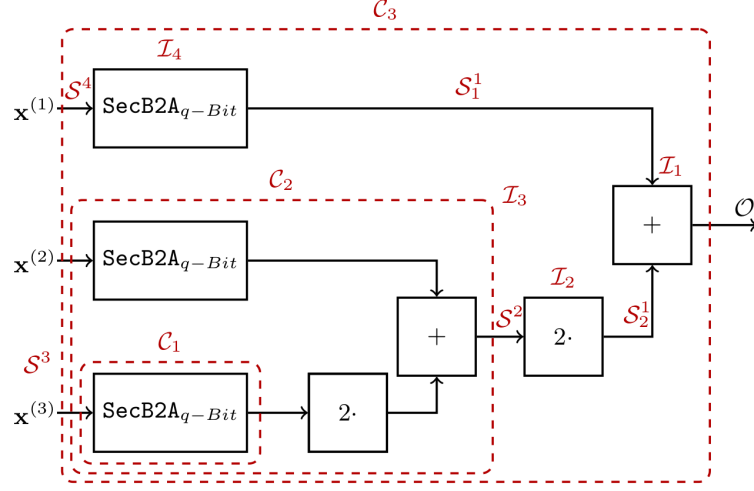
6.1.3. A new B2A_q conversion algorithm for $x \in \mathbb{F}_{2^k}$

The approach presented in the previous section, despite being efficient and simple, has the disadvantage of computing the right results only for Boolean encodings of bit values. If we consider arbitrary bit sizes, the arithmetic property in Equation 6.2 does not hold anymore, since it is specific for case of 2-bit sized values. We generalize the approach by applying the previous conversion distinctly to each input bit and combine with component-wise addition. This approach results in a complexity $\mathcal{O}(n^2 \cdot k)$, as we have k calls to $\text{SecB2A}_{q-\text{Bit}}$. The complete conversion algorithm is represented in Algorithm 6.9 and its basic structure for $k = 3$ is depicted in Figure 6.6.

Correctness

Let us assume that $2^k \leq q$. It is easy to see that for each $i = 1, \dots, n$ the output shares are of the following form

$$A_i = 2^0 \cdot \text{B2A}_{q-\text{Bit}}(x_i^{(1)}) + 2^1 \cdot \text{B2A}_{q-\text{Bit}}(x_i^{(2)}) + \dots + 2^{k-1} \cdot \text{B2A}_{q-\text{Bit}}(x_i^{(k)})$$


 Figure 6.6.: Structure of SecB2A_q in Algorithm 6.9 for $k = 3$

and therefore $\sum_{i=1}^n A_i = x \pmod q$.

Security analysis

As done for the previous algorithms, we give in the following the proof of security according to the t -SNI property, for $t \leq n-1$.

Proposition 6.6. SecB2A_q in Algorithm 6.9 is t -SNI, with $t \leq n-1$.

Proof. The proof that Algorithm 6.9 is t -SNI follows from the fact that $\text{SecB2A}_{q-\text{Bit}}$ is t -SNI and from the structure of the algorithm itself, depicted in Figure 6.6.

First, we define C_i , for $i = 1, \dots, k$, represented in Figure 6.6, where $C_1 := \text{SecB2A}_{q-\text{Bit}}$ and $C_n := \text{SecB2A}_q$. We prove the statement by induction on $i \in \mathbb{N}$.

Thanks to Proposition 6.5, C_1 is t -SNI.

We suppose now that C_i is t -SNI for all $i = 1, \dots, n-1$ and we prove that C_n is t -SNI. Let $\Omega = (\mathcal{I}, \mathcal{O})$ be the set of adversarial observations on C_n , where \mathcal{I} are the ones on the internal values and \mathcal{O} the ones on the output values, with $|\mathcal{I}| + |\mathcal{O}| \leq t$. In particular, let \mathcal{I}_1 be the set of probes on $+$, \mathcal{I}_2 the set of probes on $2 \cdot$, \mathcal{I}_3 be the set of probes on C_{n-1} and \mathcal{I}_4 be the set of probes on $\text{SecB2A}_{q-\text{Bit}}$, with $\sum_j |\mathcal{I}_j| \leq |\mathcal{I}|$.

We prove the existence of a simulator which can simulate the adversary's view by using at most $|\mathcal{I}|$ input shares. We proceed with the analysis of the circuit from right to left.

Since $+$ is a linear operation and $|\mathcal{I}_1 \cup \mathcal{O}| \leq t$, there exist two observation sets $\mathcal{S}_1^1, \mathcal{S}_2^1$ such that $|\mathcal{S}_1^1| \leq |\mathcal{I}_1 \cup \mathcal{O}|$, $|\mathcal{S}_2^1| \leq |\mathcal{I}_1 \cup \mathcal{O}|$ and the gadget can be simulated using at most $|\mathcal{S}_1^1| + |\mathcal{S}_2^1|$ shares of the inputs.

Since $2\cdot$ is a linear operation and $|I_2 \cup \mathcal{S}_2^1| \leq t$, then there exists an observation set \mathcal{S}^2 such that $|\mathcal{S}^2| \leq |I_2 \cup \mathcal{S}_2^1|$ and the gadget can be simulated using at most $|\mathcal{S}^2| + |\mathcal{S}_2^1|$ shares of the inputs.

Since, for the assumption step, \mathcal{C}_{n-1} is $t - \text{SNI}$ and moreover $|I_3 \cup \mathcal{S}^2| \leq t$, there exists an observation set \mathcal{S}^3 such that $|\mathcal{S}^3| \leq |I_3|$ and the gadget can be simulated using at most $|\mathcal{S}^3|$ shares of the inputs.

Since $\text{SecB2A}_{q-\text{Bit}}$ is $t - \text{SNI}$ and moreover $|I_4 \cup \mathcal{S}_1^1| \leq t$, there exists an observation set \mathcal{S}^4 such that $|\mathcal{S}^4| \leq |I_4|$ and the gadget can be simulated using at most $|\mathcal{S}^4|$ shares of the inputs.

By combining the steps above, we see that \mathcal{C}_n can be simulated by using a total of $|\mathcal{S}^3 \cup \mathcal{S}^4| \leq |I_3| + |I_4| \leq |I|$ input shares, completing the proof. \square

6.1.4. Performance analysis

We report in the following the performance analysis conducted in [Sch+19]. Our new conversion algorithms are compared to the cubic conversion from [Bar+18b] for a prime modulo $q = 12289$ as used in NEWHOPE. Additionally, SecB2A_q is compared to the conversions from [CGV14] and [BCZ18], since they are currently the fastest algorithms for power-of-two moduli secure at arbitrary orders

Number of operations

For comparison, the authors estimated the number of operations of the different Boolean-to-arithmetic conversions.

For the comparison of B2A_q conversions, it is considered the prime modulo from the NIST submission NEWHOPE $q = 12289$. Since the modular addition SecAddModp requires $2^k > 2q$, the authors always call SecBoolArithModp with $k' = \lceil \log_2 2q \rceil = 15$ in this evaluation. The results are summarized in Table 6.1.

Additionally, we compare our new algorithms with the state-of-the-art B2A conversions assuming a modulo of 2^k for different values of k . The resulting performances are given in Table 6.2. As expected, the algorithm of [BCZ18] outperforms all other solutions for small n . Surprisingly, however, our new conversion SecB2A_q outperforms the approach of [CGV14] for the considered values of k .

Randomness complexity

As for the randomness complexity, the authors estimated the number of required random bits for the different Boolean-to-arithmetic conversions.

As before, B2A_q conversions are initially compared considering the prime modulo from the NIST submission NEWHOPE $q = 12289$. The results are reported in Table 6.3 and

	n	2	3	4	5	6	7	8	9	10	11	16
$k = 4$	(A)	76	174	308	478	684	926	1,204	1,518	1,868	2,254	4,724
$k = 8$	(A)	156	354	624	966	1,380	1,866	2,424	3,054	3,756	4,530	9,480
	(A)	296	669	1,177	1,820	2,598	3,511	4,559	5,742	7,060	8,513	17,803
$k = 15$	(B)	765	2,451	5,617	10,722	18,225	28,585	42,261	59,712	81,397	107,775	326,125
	(C)	695	1,948	3,513	5,842	8,483	11,592	15,013	19,354	24,007	29,128	60,973

Table 6.1.: Operation count for B2A conversions with prime modulo $q = 12289$ for (A) **SecB2A_q**, (B) **SecBoolArithModp** (cubic) [Bar+18b], and (C) **SecArithBoolModp** (quadratic) [Sch+19]

again **SecB2A_q** outperforms **SecBoolArithModp**. However, the authors underline that all random samples for **SecB2A_q** are from \mathbb{F}_q , while **SecBoolArithModp** only requires $(n - 1)$ random values from \mathbb{F}_q and the remaining are sampled from $\mathbb{F}_{2^{k_1}}$, which can be more efficient depending on the RNG.

As for the cases of a power-of-two modulo, the authors estimate the number of RNG calls. Table 6.4 summarizes the resulting performances. We can see again that the algorithm of [BCZ18] outperforms all other algorithms for small n , while **SecB2A_q** gives the best performance for specific values of k and n .

6.2. Masked implementation for higher-order binomial samplers

In this section, we present two new masked binomial samplers: the first one is a generalization over the work in [Ode+18], and the second is a new bit-sliced sampler.

According to several schemes that rely on a Boolean-masked PRNG to produce uniform pseudorandomness, the inputs of the sampler algorithm are two variables (x, y) of length κ bits. We indicate with (\mathbf{x}, \mathbf{y}) their Boolean encoding. The goal of the sampler is to securely compute the difference between the Hamming weights of such encodings $\text{HW}(x) - \text{HW}(y)$ and, in line to the modulo q of the subsequent lattice operations, to produce arithmetic shares \mathbf{A} such that $\sum_i A_i = \text{HW}(x) - \text{HW}(y) \pmod{q}$. The last step can be performed employing any **B2A_q** scheme.

Our contribution is divided into two parts. We start by presenting a generalization of the first-order sampler in [Ode+18]. Then, we introduce a more efficient sampling algorithm based on bitslicing. The performances of both algorithms are compared using NEWHOPE as a case study.

	n	2	3	4	5	6	7	8	9	10	11	16
$\forall k$	(D)	15	49	123	277	591	1,225	2,499	5,053	10,167	20,401	655,251
$k = 1$	(A)	12	33	63	102	150	207	273	348	432	525	1,125
	(E)	56	135	234	374	534	721	928	1,183	1,458	1,760	3,640
$k = 4$	(A)	76	174	308	478	684	926	1,204	1,518	1,868	2,254	4,724
	(E)	134	354	636	1,052	1,530	2,098	2,728	3,520	4,374	5,318	11,248
$k = 8$	(A)	156	354	624	966	1,380	1,866	2,424	3,054	3,756	4,530	9,480
	(E)	238	646	1,172	1,956	2,858	3,934	5,128	6,636	8,262	10,062	21,392
$k = 32$	(A)	636	1,434	2,520	3,894	5,556	7,506	9,744	12,270	15,084	18,186	38,016
	(E)	862	2,398	4,388	7,380	10,826	14,950	19,528	25,332	31,590	38,526	82,256

Table 6.2.: Operation count for B2A conversions modulo 2^k for (A) SecB2A_q , (D) Bettale *et al.* [BCZ18], and (E) Coron *et al.* [CGV14]. The bold indicates where our new algorithm provides the best performance [Sch+19]

	n	2	3	4	5	6	7	8	9	10	11	16
$k = 4$	(A)	112	336	672	1,120	1,680	2,352	3,136	4,032	5,040	6,160	13,440
$k = 8$	(A)	224	672	1,344	2,240	3,360	4,704	6,272	8,064	10,080	12,320	26,880
$k = 15$	(A)	420	1,260	2,520	4,200	6,300	8,820	11,760	15,120	18,900	23,100	50,400
	(B)	1,244	4,933	12,282	24,506	42,820	68,439	102,578	146,452	201,276	268,265	828,210
	(C)	854	2,968	5,922	10,556	16,030	22,764	30,338	40,012	50,526	62,300	137,970

Table 6.3.: Required random bits for B2A conversions with $q = 12289$ for (A) SecB2A_q , (B) SecBoolArithModp (cubic) [Bar+18b], and (C) SecArithBoolModp (quadratic). Note that sampling from \mathbb{F}_q is estimated with $\lceil \log_2 q \rceil$ bits [Sch+19]

6.2.1. Generalization at high-order of [Ode+18]

Before giving our improved algorithm, we first recap the construction from [Ode+18].

The sampler uses a Boolean-masked SHAKE core [Dwo15], i.e., an algorithm that takes an arbitrarily long input string and it returns an arbitrarily long pseudorandom output string, to generate masked pseudorandom bits uniformly distributed. The inputs of the sampler are two 8-bit vectors (x, y) produced by the SHAKE, and the algorithm calculates the difference of their Hamming weights. At this point, the sampler needs to convert from Boolean masking to Arithmetic masking modulo the prime q of the lattice scheme where it is used. To this end, the authors exploit the relation in Equation 6.2.

	n	2	3	4	5	6	7	8	9	10	11	16
$\forall k$	(D)	2	7	18	41	88	183	374	757	1,524	3,059	98,286
$k = 1$	(A)	2	6	12	20	30	42	56	72	90	110	240
	(E)	9	23	41	66	95	129	167	213	263	318	663
$k = 4$	(A)	8	24	48	80	120	168	224	288	360	440	960
	(E)	15	44	83	141	209	291	383	498	623	762	1,647
$k = 8$	(A)	16	48	96	160	240	336	448	576	720	880	1,920
	(E)	23	72	139	241	361	507	671	878	1,103	1,354	2,959
$k = 32$	(A)	64	192	384	640	960	1,344	1,792	2,304	2,880	3,520	7,680
	(E)	71	240	475	841	1,273	1,803	2,399	3,158	3,983	4,906	10,831

Table 6.4.: Number of RNG calls for Boolean-to-arithmetic conversions modulo 2^k for (A) **SecB2A_q**, (D) Bettale *et al.* [BCZ18], and (E) Coron *et al.* [CGV14]. The bold indicates where our new algorithm provides the best performance [Sch+19]

The resulting masked sampler is in Algorithm 4 from [Ode+18], provided in Appendix A.

The approach mentioned above is our starting point for the construction of a generalized sampler at high-order for any modulo q , and length of the bit-vectors κ , which we give in Algorithm 6.10. Our main idea consists of first transforming each of the 2κ bits individually to an Arithmetic encoding modulo q and then summing component-wise to calculate the Hamming weight of each variable. The results are subtracted again component-wise.

Correctness

The correctness of **SecSampler₁** follows directly by the construction of the algorithm. Indeed, since at every iteration of the loop

$$\begin{aligned} \mathbf{A} = & \text{B2A}_q((\mathbf{x} \gg 0) \wedge 1) - \text{B2A}_q((\mathbf{y} \gg 0) \wedge 1) \\ & + \dots + \text{B2A}_q((\mathbf{x} \gg \kappa - 1) \wedge 1) - \text{B2A}_q((\mathbf{y} \gg \kappa - 1) \wedge 1) \end{aligned}$$

we have

$$\begin{aligned} \sum_i A_i &= \sum_i (\text{B2A}_q((x_i \gg 0) \wedge 1) - \text{B2A}_q((y_i \gg 0) \wedge 1) + \dots \\ &\quad + \text{B2A}_q((x_i \gg \kappa - 1) \wedge 1) - \text{B2A}_q((y_i \gg \kappa - 1) \wedge 1)) \\ &= \text{HW}(x) - \text{HW}(y) \pmod{q} \end{aligned}$$

Algorithm 6.10 `SecSampler1`

Input: $\mathbf{x} = (x_i)_{1 \leq i \leq n} \in \mathbb{F}_{2^\kappa}$, $\mathbf{y} = (y_i)_{1 \leq i \leq n} \in \mathbb{F}_{2^\kappa}$ such that $\bigoplus_i x_i = x$, $\bigoplus_i y_i = y$

Output: $\mathbf{A} = (A_i)_{1 \leq i \leq n} \in \mathbb{F}_q$ such that $\sum_i A_i = \text{HW}(x) - \text{HW}(y) \pmod q$

```

1:  $(A_i)_{1 \leq i \leq n} \leftarrow 0$ 
2: for  $j = 0$  to  $\kappa - 1$  do
3:    $\mathbf{B} \leftarrow \text{B2A}_q((\mathbf{x} \gg j) \wedge 1)$ 
4:    $\mathbf{C} \leftarrow \text{B2A}_q((\mathbf{y} \gg j) \wedge 1)$ 
5:    $\mathbf{A} \leftarrow \mathbf{A} + \mathbf{B} \pmod q$ 
6:    $\mathbf{A} \leftarrow \mathbf{A} - \mathbf{C} \pmod q$ 

```

Security analysis

The security of the sampler described in Algorithm 6.10 can be easily derived from its basic structure and use of $t - \text{SNI}$ gadgets.

Proposition 6.7. *`Sampler1` in Algorithm 6.10 is $t - \text{SNI}$, with $t \leq n - 1$.*

Proof. The $t - \text{SNI}$ security of both considered B2A_q algorithms, i.e., $\text{SecB2A}_{q-\text{Bit}}$ proven in Proposition 6.5 and SecArithBoolModp in Proposition 6.2, receiving independent inputs, guarantees that every loop of Algorithm 6.10 represents a $t - \text{SNI}$ gadget and therefore the output \mathbf{A} can be securely injected in the sum of the following loop. \square

6.2.2. New bit-sliced masked binomial sampler

As a first step, in our improved sampler `SecSampler2` we compute the Hamming weight of x on the Boolean encodings using bit-slicing. This allows us to significantly increase the throughput. Secondly, we directly subtract the Hamming weight of y from the result, again using bit-slicing. This strategy additionally permits to improve the performance, as the sampler only needs a single conversion algorithm. Before converting the difference and giving the final result, the sampler needs to add κ , as depicted in Algorithm 6.13. This step guarantees to correctly map the sign of the difference and to avoid that negative values are wrongly transformed. We underline that, in some cases, this operation can be significantly optimized. For instance, when $\kappa = 8$, as in `NEWHOPE`, the addition can be performed with only component-wise XOR, as described in Algorithm 6.14. Lastly, after the B2A_q conversion, we need to subtract the additional κ , in order to recover the correct result. This operation is performed component-wise on the Arithmetic shares. The full procedure is given in Algorithm 6.15. Note that, since the input variables are bit-sliced, we indicate the j^{th} bit of the l^{th} share of \mathbf{x} as $x_l^{(j)}$.

Algorithm 6.11 SecBitAdd

Input: $\mathbf{x} = (x_i)_{1 \leq i \leq n} \in \mathbb{F}_{2^\kappa}$ such that $\bigoplus_i x_i = x$, $\lambda = \lceil \log_2(\kappa + 1) \rceil + 1$

Output: $\mathbf{z} = (z_i)_{1 \leq i \leq n} \in \mathbb{F}_{2^\lambda}$ such that $\bigoplus_i z_i = \text{HW}(x)$

- 1: $(t_i)_{1 \leq i \leq n} \leftarrow 0$
 - 2: $(z_i)_{1 \leq i \leq n} \leftarrow 0$
 - 3: **for** $j = 1$ to κ **do**
 - 4: $\mathbf{t}^{(1)} \leftarrow \mathbf{z}^{(1)} \oplus \mathbf{x}^{(j)}$
 - 5: $\mathbf{w} \leftarrow \mathbf{x}^{(j)}$
 - 6: **for** $l = 2$ to λ **do**
 - 7: $\mathbf{w} \leftarrow \text{SecAnd}(\mathbf{w}, \mathbf{z}^{(l-1)})$
 - 8: $\mathbf{t}^{(l)} \leftarrow \mathbf{z}^{(l)} \oplus \mathbf{w}$
 - 9: $\mathbf{z} \leftarrow \mathbf{t}$
-

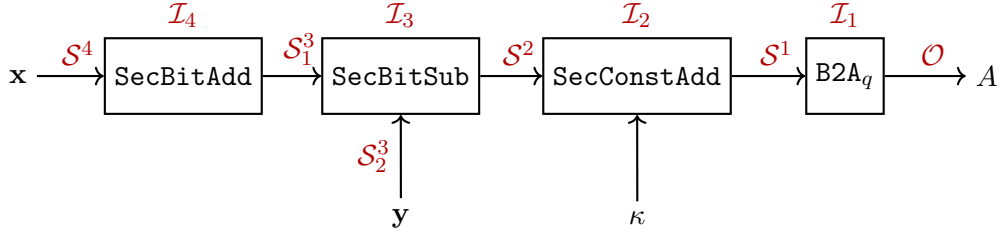


Figure 6.7.: Structure of SecSampler_2 in Algorithm 6.15 (lines 1-4)

Correctness

The correctness of SecSampler_2 is easy to show.

$$\begin{aligned}
 \sum_i A_i &= \text{B2A}_q(\text{SecConstAdd}(\text{SecBitSub}(\text{SecBitAdd}(\mathbf{x}), \mathbf{y}), \kappa)) - \kappa \\
 &= \text{B2A}_q(\text{SecConstAdd}(\text{SecBitSub}(\text{HW}(x), \mathbf{y}), \kappa)) - \kappa \\
 &= \text{B2A}_q(\text{SecConstAdd}(\text{HW}(x) - \text{HW}(y), \kappa)) - \kappa \\
 &= \text{B2A}_q(\text{HW}(x) - \text{HW}(y) + \kappa) - \kappa = \text{HW}(x) - \text{HW}(y) \pmod{q}.
 \end{aligned}$$

Security analysis

Before proving the security of SecSampler_2 in Algorithm 6.15, we briefly summarize the security properties which its subroutines satisfy.

First we show that SecBitAdd in Algorithm 6.11 is $t - \text{NI}$ and we start the analysis by focusing on its structure. We recall that SecAnd [CGV14] is $t - \text{SNI}$ and it receives at every iteration independent inputs (line 7). The output of each SecAnd is added with

Algorithm 6.12 SecBitSub

Input: $\mathbf{z} = (z_i)_{1 \leq i \leq n} \in \mathbb{F}_{2^\lambda}, \mathbf{y} = (y_i)_{1 \leq i \leq n} \in \mathbb{F}_{2^\kappa}$ such that $\bigoplus_i z_i = z$ and $\bigoplus_i y_i = y$, $\lambda = \lceil \log_2(\kappa + 1) \rceil + 1$

Output: $\mathbf{z} = (z_i)_{1 \leq i \leq n} \in \mathbb{F}_{2^\lambda}$ such that $\bigoplus_i z_i = z - \text{HW}(y)$

```

1:  $(t_i)_{1 \leq i \leq n} \leftarrow 0$ 
2: for  $j = 1$  to  $\kappa$  do
3:    $\mathbf{t}^{(1)} \leftarrow \mathbf{z}^{(1)} \oplus \mathbf{y}^{(j)}$ 
4:    $\mathbf{w} \leftarrow \mathbf{y}^{(j)}$ 
5:   for  $l = 2$  to  $\lambda$  do
6:      $\mathbf{u} \leftarrow \mathbf{z}^{(l-1)}$ 
7:      $u_1 \leftarrow \neg u_1$ 
8:      $\mathbf{w} \leftarrow \text{SecAnd}(\mathbf{w}, \mathbf{u})$ 
9:      $\mathbf{t}^{(l)} \leftarrow \mathbf{z}^{(l)} \oplus \mathbf{w}$ 
10:   $\mathbf{z} \leftarrow \mathbf{t}$ 
    
```

Algorithm 6.13 SecConstAdd

Input: $\mathbf{x} = (x_i)_{1 \leq i \leq n} \in \mathbb{F}_{2^\lambda}$, $\lambda = \lceil \log_2(\kappa + 1) \rceil + 1$

Output: $\mathbf{y} = (y_i)_{1 \leq i \leq n} \in \mathbb{F}_{2^\lambda}$ such that $\bigoplus_i y_i = x + \kappa$

```

1:  $(t_i)_{1 \leq i \leq n} \leftarrow (\kappa, 0, \dots, 0)$ 
2:  $\mathbf{t} \leftarrow \text{RefreshXOR}(\mathbf{t}, \lambda)$ 
3:  $\mathbf{y} \leftarrow \text{SecAdd}(\mathbf{x}, \mathbf{t})$ 
    
```

a XOR to a value independent from its inputs (line 8), therefore the entire inner loop (lines 6-9) represents a $t - \text{SNI}$ gadget. This is recursively composed in the outer loop (lines 3-11) providing the outputs $(\mathbf{t}^{(2)}, \dots, \mathbf{t}^{(\lambda)})$ and preserving the $t - \text{SNI}$ property. Additionally, at every iteration of the outer loop, $\mathbf{x}^{(j)}$ is added with a XOR to $\mathbf{z}^{(1)}$ (line 4), resulting in the output $\mathbf{t}^{(1)} = \mathbf{x}^{(1)} + \dots + \mathbf{x}^{(\kappa)}$. Let us suppose an attacker probes a set of t_1 values \mathcal{P}_1 on the shares $(\mathbf{t}^{(2)}, \dots, \mathbf{t}^{(\lambda)})$ or on the internal values produced during the concatenation of the inner loop, and a set of t_2 values \mathcal{P}_2 on the computation of $\mathbf{t}^{(1)}$, with $t_1 + t_2 \leq t$. In particular let t_1^O, t_2^O be the probes on the output values and t_1^I, t_2^I the ones on the internals, with $t_1^O + t_1^I = t_1$ and $t_2^I + t_2^O = t_2$. The $t - \text{SNI}$ of the inner loop guarantees that every value in \mathcal{P}_1 can be simulated by using at most t_1^I shares of the inputs. On the other hand, because of the linearity of the computation of $\mathbf{t}^{(1)}$, the probes in \mathcal{P}_2 can be simulated using at most $t_2^I + t_2^O$ shares of the input. Therefore, by Definition 2.4, SecBitAdd is $t - \text{NI}$.

Now, since SecBitSub in Algorithm 6.12 follows the same procedure as Algorithm 6.11, with the exception of Lines 6 and 7, which simply add a negation to the interested value, then it is $t - \text{NI}$ as well.

As for SecConstAdd in Algorithm 6.13, from [Bar+18b] we know that RefreshXOR

Algorithm 6.14 `SecConstAdd` (optimized for $\kappa = 8$)

Input: $\mathbf{x} = (x_i)_{1 \leq i \leq n} \in \mathbb{F}_{2^\lambda}$

Output: $\mathbf{y} = (y_i)_{1 \leq i \leq n} \in \mathbb{F}_{2^\lambda}$ such that $\bigoplus_i y_i = x + 8$

- 1: $\mathbf{y} \leftarrow \mathbf{x}$
 - 2: $\mathbf{y}^{(5)} \leftarrow \mathbf{y}^{(5)} \oplus \mathbf{y}^{(4)}$
 - 3: $y_1^{(4)} \leftarrow y_1^{(4)} \oplus 1$
-

Algorithm 6.15 `SecSampler2`

Input: $\mathbf{x} = (x_i)_{1 \leq i \leq n} \in \mathbb{F}_{2^\kappa}$, $\mathbf{y} = (y_i)_{1 \leq i \leq n} \in \mathbb{F}_{2^\kappa}$, κ , such that $\bigoplus_i x_i = x$, $\bigoplus_i y_i = y$

Output: $\mathbf{A} = (A_i)_{1 \leq i \leq n} \in \mathbb{F}_q$ such that $\sum_i A_i = \text{HW}(x) - \text{HW}(y) \pmod q$

- 1: $\mathbf{z} \leftarrow \text{SecBitAdd}(\mathbf{x})$
 - 2: $\mathbf{z} \leftarrow \text{SecBitSub}(\mathbf{z}, \mathbf{y})$
 - 3: $\mathbf{z} \leftarrow \text{SecConstAdd}(\mathbf{z}, \kappa)$
 - 4: $\mathbf{A} \leftarrow \text{B2A}_q(\mathbf{z})$
 - 5: $A_1 \leftarrow A_1 - \kappa \pmod q$
-

is $t - \text{SNI}$ and `SecAdd` is $t - \text{NI}$. Therefore it is easy to see that the composition of them, as it appears in Algorithm 6.13, is $t - \text{NI}$. Regarding the optimized version of `SecConstAdd` in Algorithm 6.14, here the security comes directly from the fact that the algorithm is linear.

Proposition 6.8. `SecSampler2` in Algorithm 6.15 is $t - \text{SNI}$, with $t \leq n - 1$.

Proof. Before proceeding with the proof, we point out that `SecSampler2` is given by the circuit in Figure 6.7 with the addition of a share-wise sum between the output share A_1 and the public value $-\kappa$ (line 5 of Algorithm 6.15). Since the simulation of this value depends only on the simulation of A_1 , the security of `SecSampler2` is not influenced by this additional operation and it corresponds to the one of the algorithm in Figure 6.7.

Let $\Omega = (\mathcal{I}, \mathcal{O})$ be the set of adversarial observations on the circuit in Figure 6.7, where \mathcal{I} are the ones on the internal values and \mathcal{O} on the output shares, with $|\mathcal{I}| + |\mathcal{O}| \leq t$. In particular, let \mathcal{I}_1 be the set of probes on `B2Aq`, \mathcal{I}_2 on `SecConstAdd`, \mathcal{I}_3 on `SecBitSub` and \mathcal{I}_4 on `SecBitAdd`, with $\sum_j |\mathcal{I}_j| \leq |\mathcal{I}|$.

We prove the existence of a simulator which simulates the adversary's view by using at most $|\mathcal{I}|$ input shares, analyzing the circuit from right to left.

Since `B2Aq` is $t - \text{SNI}$, there exists an observation set \mathcal{S}^1 such that $|\mathcal{S}^1| \leq |\mathcal{I}_1|$ and the gadget can be simulated using at most $|\mathcal{S}^1|$ shares of its input.

Since `SecConstAdd` is $t - \text{NI}$ and $|\mathcal{I}_2 \cup \mathcal{S}^1| \leq t$, then there exists an observation set \mathcal{S}^2 such that $|\mathcal{S}^2| \leq |\mathcal{I}_2 \cup \mathcal{S}^1|$ and the gadget can be simulated using at most $|\mathcal{S}^2|$ shares of the inputs.

Since **SecBitSub** is $t - \text{NI}$ and $|\mathcal{I}_3 \cup \mathcal{S}^2| \leq t$, then there exist two observation sets $\mathcal{S}_1^3, \mathcal{S}_2^3$ such that $|\mathcal{S}_1^3| \leq |\mathcal{I}_3 \cup \mathcal{S}^2|$, $|\mathcal{S}_2^3| \leq |\mathcal{I}_3 \cup \mathcal{S}^2|$ and the gadget can be simulated using at most $|\mathcal{S}_1^3| + |\mathcal{S}_2^3|$ shares of the inputs.

Since **SecBitAdd** is $t - \text{NI}$ and $|\mathcal{I}_4 \cup \mathcal{S}_1^3| \leq t$, then there exists an observation set \mathcal{S}^4 such that $|\mathcal{S}^4| \leq |\mathcal{I}_4 \cup \mathcal{S}_1^3|$ and the gadget can be simulated using at most $|\mathcal{S}^4|$ shares of the inputs.

By combining the steps above, we see that **SecSampler₂** can be simulated by using $|\mathcal{S}^4| \leq |\mathcal{I}_4| + |\mathcal{S}_1^3| \leq |\mathcal{I}_4| + |\mathcal{I}_3| + |\mathcal{S}^2| \leq |\mathcal{I}_4| + |\mathcal{I}_3| + |\mathcal{I}_2| + |\mathcal{S}^1| \leq |\mathcal{I}_4| + |\mathcal{I}_3| + |\mathcal{I}_2| + |\mathcal{I}_1| \leq |\mathcal{I}|$ input shares, proving that it is $t - \text{SNI}$. \square

6.3. Case study: NewHope

To have a concrete idea of the performance of our new samplers, we report the performance evaluations provided in [Sch+19], where the authors conducted a case study using the parameters of the NIST submission NEWHOPE. The length of the bit-vectors is set to $\kappa = 8$ and the prime to $q = 12289$.

Both sampling approaches are evaluated with the **B2A_q** conversion algorithms **SecB2A_q** and **SecArithBoolModp** (quadratic), presented in Section 6.1. The latter is instantiated with $k' = \lceil \log_2 2q \rceil = 15$.

The implementations are performed on a 32-bit ARM Cortex-M4F microcontroller embedded in an STM32F4 DISCOVERY board with 1 Mbyte of flash memory, 192 kbyte of RAM, a floating-point unit (FPU), and a true random number generator (TRNG). The results can be seen in Table 6.5.

We point out that our new sampler **SecSampler₂** significantly improves **SecSampler₁** regardless the conversion algorithm adopted and it additionally outperforms the first-order scheme from Oder *et al.* [Ode+18].

6.4. Conclusions

In this chapter, we presented the first masked binomial sampler provable secure at high-order against a power analysis attacker. Additionally, we showed a masked bit-sliced sampler, which provides secure and efficient sampling even at large protection orders. The algorithms rely on new conversion algorithms from Boolean to Arithmetic masking schemes for prime moduli, which improve previous algorithms significantly for the relevant parameters. In particular, we presented two new conversion schemes that transform Boolean shares to Arithmetic shares with arbitrary moduli and security orders. The first one has asymptotic complexity $\mathcal{O}(n^2 \log k)$, where n is the number of shares and k the bit-size of the conversion, outperforming the algorithms in its category. The second conversion, instead, shows a notable performance improvement for relevant bit sizes.

	n	2	3	4	5
	[Ode+18]	3,637	-	-	-
SecSampler ₁	(C)	271,423	638,315	1,076,155	1,758,184
	(A)	6,145	13,913	24,397	37,880
SecSampler ₂	(C)	17,564	40,977	68,914	112,402
	(A)	2,649	5,573	9,462	14,587

Table 6.5.: Cycle counts for masked sampling of one binomial distributed coefficient using the B2A conversions (A) SecB2A_q and (C) SecBoolArithModp (quadratic). We excluded the generation of the input bit vectors for better comparison as this is a constant overhead for all approaches [Sch+19]

For specific parameters, e.g., $n \geq 11$ for $k = 32$, it also improves previous works for power-of-two moduli, and it is, therefore, suitable for the use in symmetric cryptography as well.

Since our proposed schemes support arbitrary moduli, they can be applied to a broad class of lattice-based constructions, like NEWHOPE, LIMA, Saber, Kyber, HILA5, or Ding Key Exchange. While the NIST standardization process is ongoing, this peculiarity makes them an essential contribution in order to compare the submitted schemes and to have a more precise analysis of them. In a more general picture, our study advances the understanding of the cost that masking post-quantum cryptography requires.

7. Conclusion

Finally, we conclude our work by summarizing our advancements in the field of provable countermeasures against power analysis attacks and discussing some future research possibilities.

The first two chapters presenting the new contributions of this thesis focused on the challenge of improving efficiency in masking schemes, which we carried out by proposing strategies to reduce the randomness cost in masked implementations. We developed improvements for Boolean masking and Inner Product masking. In the first case, we proposed a method to recycle part of the internal randomness used among gadgets. We proposed new notions that formalize security for gadgets that share part of their randomness, and we explicitly provided a method to design multiplication schemes fulfilling such security requirements. In the second case, we improved the efficiency of prior multiplication schemes proposing new algorithms that are conceptually close to the schemes of Ishai *et al.* [ISW03] and show a reduced randomness cost. Additionally, we optimized a specific composition pattern common in the AES S-box, which limits further the use of random elements in some masked implementations.

By studying both Boolean and Inner Product masking, we remarked that there is a trade-off between efficiency and security level in masked implementations. The simple Boolean masking provides easy algorithmic design and relatively high performance. On the other hand, the Inner Product masking, characterized by a more complicated algebraic structure, shows less evidence of leakage in practice, and, in some cases, a security order amplification, but at the cost of higher performance overhead. The reduction in the performance overhead that we provided for the Inner Product masking contributes to reduce this gap and shows that Inner Product masking could be a valid alternative to Boolean masking.

In the subsequent chapter, we considered the problem of defining a suitable model that formally represents the local security and composability of masked gadgets in the presence of physical defaults. The most popular model to prove the security of masked algorithms is currently the probing model [ISW03]. While it provides a simple way to examine the security of algorithms, this model is based on idealistic assumptions and does not capture physical peculiarities, like glitches or transitional leakages, leading to inconsistency between the theoretical security of an algorithm and the practical security of its implementation. To fill this gap, we defined the robust probing model, which allows

a designer to analyze and demonstrate security guarantees against different kinds of physical defaults, and we used it to prove that the ISW multiplication gadget is probing secure in the presence of glitches in 2 cycles, and it shows composability guarantees.

As a last contribution, we worked more concretely on building masked implementations of algorithms used in lattice-based cryptosystems. In this context, we gave two main contributions. The first one is a new algorithm for the conversion from Boolean to Arithmetic masking at arbitrary moduli. The second one is the first masked binomial sampler provable secure at high-order against a power analysis attacker. Both schemes can be applied to a wide class of lattice-based constructions, like NEWHOPE, LIMA, Saber, Kyber, HILA5, or Ding Key Exchange, which are part of the submissions to the NIST post-quantum standardization process. This broad applicability to lattice-based constructions makes our work fundamental for a comprehensive understanding of their differences because it contributes to estimate the overhead cost that masking requires.

Future directions

This work offers several future research directions and raises many interesting open problems.

In the field of randomness reuse, it could be interesting analyzing different security requirements that allow the construction of multiplication schemes with shared randomness for order higher than 3. Another direction in which it would be worth investigating is a randomness reusing strategy following a different logic than the one proposed in this work. For instance, an alternative could be to cyclically reuse part of the encoding of one gadget as randomness of another gadget.

With the development of the robust probing model, we did the first step in closing the gap between theoretic and practical security. Still, our model can be improved to capture more realistic adversarial strategies. For instance, a possibility of improvement could be considering not only the scenario of passive attackers but also the one of active attackers. Such a model could also be used to study a randomness reusing strategy that exhibits security in a setting closer to reality.

In the field of masking lattice-based cryptography, a lot can still be done. This thesis mainly covers a high-order masked implementation of the binomial sampler, which is common in lattice-based constructions. However, many other components lack a protected implementation, for example, in order to mask the whole NEWHOPE key exchange, we still need to design a masked version of the encoding and decoding algorithms.

A. Appendix

A.1. Additional algorithms used in Chapter 6

Algorithm A.1 SecAddModp [Bar+18b]

Input: $\mathbf{x} = (x_i)_{1 \leq i \leq n} \in \mathbb{F}_{2^k}$, $\mathbf{y} = (y_i)_{1 \leq i \leq n} \in \mathbb{F}_{2^k}$ such that $\bigoplus_i x_i = x$, $\bigoplus_i y_i = y$

Output: $\mathbf{z} = (z_i)_{1 \leq i \leq n} \in \mathbb{F}_{2^k}$ with $2^k > 2q$ such that $\bigoplus_i z_i = x + y \pmod q$

- 1: $(p_i)_{1 \leq i \leq n} \leftarrow (2^k - p, 0, \dots, 0)$
 - 2: $\mathbf{s} \leftarrow \text{SecAdd}(\mathbf{x}, \mathbf{y})$
 - 3: $\mathbf{s}' \leftarrow \text{SecAdd}(\mathbf{s}, \mathbf{p})$
 - 4: $\mathbf{b} \leftarrow \mathbf{s} \gg (k - 1)$
 - 5: $\mathbf{c} \leftarrow \text{RefreshXOR}(\mathbf{b}, 1)$
 - 6: $\mathbf{z} \leftarrow \text{SecAnd}(\mathbf{s}, \tilde{\mathbf{c}})$
 - 7: $\mathbf{c} \leftarrow \text{RefreshXOR}(\mathbf{b}, 1)$
 - 8: $\mathbf{z} \leftarrow \mathbf{z} \oplus \text{SecAnd}(\mathbf{s}', \neg \tilde{\mathbf{c}})$
-

Algorithm A.2 SecAdd [Bar+18b]

Input: $\mathbf{x} = (x_i)_{1 \leq i \leq n} \in \mathbb{F}_{2^k}$, $\mathbf{y} = (y_i)_{1 \leq i \leq n} \in \mathbb{F}_{2^k}$ such that $\bigoplus_i x_i = x$, $\bigoplus_i y_i = y$

Output: $\mathbf{z} = (z_i)_{1 \leq i \leq n} \in \mathbb{F}_{2^k}$ such that $\bigoplus_i z_i = x + y \pmod{2^k}$

```

1:  $\mathbf{p} \leftarrow \mathbf{x} \oplus \mathbf{y}$ 
2:  $\mathbf{g} \leftarrow \text{SecAnd}(\mathbf{x}, \mathbf{y})$ 
3: for  $j = 1$  to  $W = \lceil \log_2(k-1) \rceil - 1$  do
4:    $\text{pow} \leftarrow 2^{j-1}$ 
5:    $\mathbf{a} \leftarrow \mathbf{g} \ll (\text{pow})$ 
6:    $\mathbf{a} \leftarrow \text{SecAnd}(\mathbf{a}, \mathbf{p})$ 
7:    $\mathbf{g} \leftarrow \mathbf{g} \oplus \mathbf{a}$ 
8:    $\mathbf{a}' \leftarrow \mathbf{p} \ll (\text{pow})$ 
9:    $\mathbf{a}' \leftarrow \text{RefreshXOR}(\mathbf{a}', k)$ 
10:   $\mathbf{p} \leftarrow \text{SecAnd}(\mathbf{p}, \mathbf{a}')$ 
11:  $\mathbf{a} \leftarrow \mathbf{g} \ll (2^W)$ 
12:  $\mathbf{a} \leftarrow \text{SecAnd}(\mathbf{a}, \mathbf{p})$ 
13:  $\mathbf{g} \leftarrow \mathbf{g} \oplus \mathbf{a}$ 
14:  $\mathbf{z} \leftarrow \mathbf{x} \oplus \mathbf{y} \oplus (\mathbf{g} \ll 1)$ 

```

Algorithm A.3 SecAnd [CGV14]

Input: $\mathbf{x} = (x_i)_{1 \leq i \leq n} \in \mathbb{F}_{2^k}$, $\mathbf{y} = (y_i)_{1 \leq i \leq n} \in \mathbb{F}_{2^k}$ such that $\bigoplus_i x_i = x$, $\bigoplus_i y_i = y$

Output: $\mathbf{z} = (z_i)_{1 \leq i \leq n} \in \mathbb{F}_{2^k}$ such that $\bigoplus_i z_i = x \wedge y$

```

1:  $\mathbf{z} \leftarrow \mathbf{x} \wedge \mathbf{y}$ 
2: for  $i = 1$  to  $n-1$  do
3:   for  $j = 1+i$  to  $n$  do
4:      $r_{i,j} \xleftarrow{\$} \mathbb{F}_{2^k}$ 
5:      $r_{j,i} \leftarrow (x_i \wedge y_j) \oplus r_{i,j}$ 
6:      $r_{j,i} \leftarrow r_{j,i} \oplus (x_j \wedge y_i)$ 
7:      $z_i \leftarrow z_i \oplus r_{i,j}$ 
8:      $z_j \leftarrow z_j \oplus r_{j,i}$ 

```

Algorithm A.4 FullXOR [CGV14]

Input: $\mathbf{x} = (x_i)_{1 \leq i \leq n} \in \mathbb{F}_{2^k}$ such that $\bigoplus_i x_i = x$

Output: x

```

1:  $\mathbf{y} \leftarrow \text{FullRefreshXOR}(\mathbf{x}, k)$ 
2:  $x \leftarrow y_1$ 
3: for  $i = 2$  to  $n$  do
4:    $x \leftarrow x \oplus y_i$ 

```

Algorithm A.5 RefreshXOR [Bar+18b]

Input: $\mathbf{x} = (x_i)_{1 \leq i \leq n} \in \mathbb{F}_{2^{k_1}}$ such that $\bigoplus_i x_i = x$, bit size k_2

Output: $\mathbf{y} = (y_i)_{1 \leq i \leq n} \in \mathbb{F}_{2^{k_2}}$ such that $\bigoplus_i y_i = x$

```

1:  $\mathbf{y} \leftarrow \mathbf{x}$ 
2: for  $i = 1$  to  $n - 1$  do
3:   for  $j = 1 + i$  to  $n$  do
4:      $r \xleftarrow{\$} \mathbb{F}_{2^{k_2}}$ 
5:      $y_i \leftarrow y_i \oplus r$ 
6:      $y_j \leftarrow y_j \oplus r$ 

```

Algorithm A.6 FullRefreshXOR [Bar+18b]

Input: $\mathbf{x} = (x_i)_{1 \leq i \leq n} \in \mathbb{F}_{2^{k_1}}$ such that $\bigoplus_i x_i = x$, bit size k_2

Output: $\mathbf{y} = (y_i)_{1 \leq i \leq n} \in \mathbb{F}_{2^{k_2}}$ such that $\bigoplus_i y_i = x$

```

1:  $\mathbf{y} \leftarrow \mathbf{x}$ 
2: for  $i = 1$  to  $n$  do
3:   for  $j = 2$  to  $n$  do
4:      $r \xleftarrow{\$} \mathbb{F}_{2^{k_2}}$ 
5:      $y_1 \leftarrow y_1 \oplus r$ 
6:      $y_j \leftarrow y_j \oplus r$ 

```

Algorithm A.7 SecMul (based on SecAnd)

Input: $\mathbf{A} = (A_i)_{1 \leq i \leq n} \in \mathbb{F}_q$, $\mathbf{B} = (B_i)_{1 \leq i \leq n} \in \mathbb{F}_q$ such that $\sum_i A_i = x \pmod q$, $\sum_i B_i \leftarrow y \pmod q$

Output: $\mathbf{C} = (C_i)_{1 \leq i \leq n} \in \mathbb{F}_q$ such that $\sum_i C_i = x + y \pmod q$

```

1:  $\mathbf{C} = \mathbf{A} \cdot \mathbf{B} \pmod q$ 
2: for  $i = 1$  to  $n - 1$  do
3:   for  $j = 1 + i$  to  $n$  do
4:      $R_{i,j} \xleftarrow{\$} \mathbb{F}_q$ 
5:      $R_{j,i} \leftarrow (A_i \cdot B_j) + R_{i,j} \pmod q$ 
6:      $R_{j,i} \leftarrow R_{j,i} + (A_j \cdot B_i) \pmod q$ 
7:      $C_i \leftarrow C_i - R_{i,j} \pmod q$ 
8:      $C_j \leftarrow C_j + R_{j,i} \pmod q$ 

```

Bibliography

- [Agr+03] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi. “The EM Side-Channel(s)”. In: *CHES*. LNCS. Springer, 2003, pp. 29–45. DOI: 10.1007/3-540-36400-5_4.
- [AGV09] A. Akavia, S. Goldwasser, and V. Vaikuntanathan. “Simultaneous Hardcore Bits and Cryptography against Memory Attacks”. In: *TCC*. LNCS. Springer, 2009, pp. 474–495. DOI: 10.1007/978-3-642-00457-5_28.
- [AJS16] E. Alkim, P. Jakubeit, and P. Schwabe. “NewHope on ARM Cortex-M”. In: *SPACE*. Cham: Springer International Publishing, 2016, pp. 332–349. DOI: 10.1007/978-3-319-49445-6_19.
- [Ald+19] A. C. Aldaya, C. P. García, L. M. A. Tapia, and B. B. Brumley. “Cache-Timing Attacks on RSA Key Generation”. In: *TCHES* 4 (2019), pp. 213–242. DOI: 10.13154/tches.v2019.i4.213-242.
- [Alk+] E. Alkim, R. Avanzi, J. Bos, L. Ducas, A. de la Piedra, T. Pöppelmann, P. Schwabe, and D. Stebila. *NewHope Algorithm Specifications and Supporting Documentation*. Available at https://newhopecrypto.org/data/NewHope_2017_12_21.pdf.
- [Alk+16] E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe. “Post-quantum Key Exchange - A New Hope”. In: *USENIX Security 16*. USENIX Association, 2016, pp. 327–343.
- [Ava+17] R. Avanzi et al. *CRYSTALS-Kyber*. Tech. rep. National Institute of Standards and Technology, 2017.
- [Bal+12] J. Balasch, S. Faust, B. Gierlichs, and I. Verbauwhede. “Theory and Practice of a Leakage Resilient Masking Scheme”. In: *ASIACRYPT*. LNCS. Springer, 2012, pp. 758–775. DOI: 10.1007/978-3-642-34961-4_45.
- [Bal+14] J. Balasch, B. Gierlichs, V. Grosso, O. Reparaz, and F. Standaert. “On the Cost of Lazy Engineering for Masked Software Implementations”. In: *CARDIS*. LNCS. Springer, 2014, pp. 64–81. DOI: 10.1007/978-3-319-16763-3_5.
- [Bal+15] J. Balasch, B. Gierlichs, O. Reparaz, and I. Verbauwhede. “DPA, Bitslicing and Masking at 1 GHz”. In: *CHES*. LNCS. Springer, 2015, pp. 599–619. DOI: 10.1007/978-3-662-48324-4_30.

- [Bal+17] J. Balasch, S. Faust, B. Gierlichs, C. Paglialonga, and F.-X. Standaert. “Consolidating Inner Product Masking”. In: *ASIACRYPT*. LNCS. Springer, 2017, pp. 724–754. DOI: 10.1007/978-3-319-70694-8_25.
- [Bar+15a] G. Barthe, S. Belaïd, F. Dupressoir, P. Fouque, and B. Grégoire. “Compositional Verification of Higher-Order Masking: Application to a Verifying Masking Compiler”. In: *IACR Cryptology ePrint Archive* (2015), p. 506.
- [Bar+15b] G. Barthe, S. Belaïd, F. Dupressoir, P.-A. Fouque, B. Grégoire, and P.-Y. Strub. “Verified Proofs of Higher-Order Masking”. In: *EUROCRYPT*. LNCS. Springer, 2015, pp. 457–485. DOI: 10.1007/978-3-662-46800-5_18.
- [Bar+16] G. Barthe, S. Belaïd, F. Dupressoir, P.-A. Fouque, B. Grégoire, P.-Y. Strub, and R. Zucchini. “Strong Non-Interference and Type-Directed Higher-Order Masking”. In: *ACM CCS 16*. ACM Press, 2016, pp. 116–129. DOI: 10.1145/2976749.2978427.
- [Bar+17] G. Barthe, F. Dupressoir, S. Faust, B. Grégoire, F.-X. Standaert, and P.-Y. Strub. “Parallel Implementations of Masking Schemes and the Bounded Moment Leakage Model”. In: *EUROCRYPT*. LNCS. Springer, 2017, pp. 535–566. DOI: 10.1007/978-3-319-56620-7_19.
- [Bar+18a] G. Barthe, S. Belaïd, T. Espitau, P. Fouque, B. Grégoire, M. Rossi, and M. Tibouchi. “Masking the GLP Lattice-Based Signature Scheme at Any Order”. In: *EUROCRYPT (2)*. Springer, 2018. DOI: 10.1007/978-3-319-78375-8_12.
- [Bar+18b] G. Barthe, S. Belaïd, T. Espitau, P.-A. Fouque, B. Grégoire, M. Rossi, and M. Tibouchi. “Masking the GLP Lattice-Based Signature Scheme at Any Order”. In: *EUROCRYPT*. LNCS. Springer, 2018, pp. 354–384. DOI: 10.1007/978-3-319-78375-8_12.
- [BBD09] D. J. Bernstein, J. Buchmann, and E. Dahmen. *Post-quantum cryptography*. Mathematics and Statistics Springer-11649; ZDB-2-SMA. Springer, 2009.
- [BCZ18] L. Bettale, J.-S. Coron, and R. Zeitoun. “Improved High-Order Conversion From Boolean to Arithmetic Masking”. In: *TCHES 2* (2018), pp. 22–45. DOI: 10.13154/tches.v2018.i2.22-45.
- [BDL97] D. Boneh, R. A. DeMillo, and R. J. Lipton. “On the Importance of Checking Cryptographic Protocols for Faults (Extended Abstract)”. In: *EUROCRYPT’97*. LNCS. Springer, 1997, pp. 37–51. DOI: 10.1007/3-540-69053-0_4.
- [Bel15] S. Belaïd. “Security of Cryptosystems Against Power-Analysis Attacks. (Sécurité des cryptosystèmes contre les attaques par analyse de courant)”. PhD thesis. École Normale Supérieure, Paris, France, 2015.

-
- [Bel+16] S. Belaïd, F. Benhamouda, A. Passelègue, E. Prouff, A. Thillard, and D. Vergnaud. “Randomness Complexity of Private Circuits for Multiplication”. In: *EUROCRYPT*. LNCS. Springer, 2016, pp. 616–648. DOI: 10.1007/978-3-662-49896-5_22.
- [Bel+17] S. Belaïd, F. Benhamouda, A. Passelègue, E. Prouff, A. Thillard, and D. Vergnaud. “Private Multiplication over Finite Fields”. In: *CRYPTO*. LNCS. Springer, 2017, pp. 397–426. DOI: 10.1007/978-3-319-63697-9_14.
- [BFG15] J. Balasch, S. Faust, and B. Gierlichs. “Inner Product Masking Revisited”. In: *EUROCRYPT*. LNCS. Springer, 2015, pp. 486–510. DOI: 10.1007/978-3-662-46800-5_19.
- [BGW88] M. Ben-Or, S. Goldwasser, and A. Wigderson. “Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation (Extended Abstract)”. In: *ACM STOC*. ACM Press, 1988, pp. 1–10. DOI: 10.1145/62212.62213.
- [Bil+13] B. Bilgin, J. Daemen, V. Nikov, S. Nikova, V. Rijmen, and G. V. Assche. “Efficient and First-Order DPA Resistant Implementations of Keccak”. In: *CARDIS*. Lecture Notes in Computer Science. Springer, 2013, pp. 187–199. DOI: 10.1007/978-3-319-08302-5_13.
- [Bil+14] B. Bilgin, B. Gierlichs, S. Nikova, V. Nikov, and V. Rijmen. “Higher-Order Threshold Implementations”. In: *ASIACRYPT 2014, Part II*. LNCS. Springer, 2014, pp. 326–343. DOI: 10.1007/978-3-662-45608-8_18.
- [Bil15] B. Bilgin. *Threshold Implementations: A Countermeasure Against Higher-Order Differential Power Analysis*. PhD Thesis, KU Leuven (Belgium) and U Twente (The Netherlands). 2015.
- [Bir+17] A. Biryukov, D. Dinu, Y. L. Corre, and A. Udovenko. “Optimal First-Order Boolean Masking for Embedded IoT Devices”. In: *CARDIS*. Springer, 2017, pp. 22–41. DOI: 10.1007/978-3-319-75208-2_2.
- [Blo+18] R. Bloem, H. Groß, R. Iusupov, B. Könighofer, S. Mangard, and J. Winter. “Formal Verification of Masked Hardware Implementations in the Presence of Glitches”. In: *EUROCRYPT*. LNCS. Springer, 2018, pp. 321–353. DOI: 10.1007/978-3-319-78375-8_11.
- [BM16] G. Bertoni and M. Martinoli. “A Methodology for the Characterisation of Leakages in Combinatorial Logic”. In: *SPACE*. LNCS. Springer, 2016, pp. 363–382. DOI: 10.1007/978-3-319-49445-6_21.

- [Bos+18] J. W. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé. “CRYSTALS - Kyber: A CCA-Secure Module-Lattice-Based KEM”. In: *IEEE European Symposium on Security and Privacy, EuroS&P*. 2018, pp. 353–367.
- [Bra+10] Z. Brakerski, Y. T. Kalai, J. Katz, and V. Vaikuntanathan. “Overcoming the Hole in the Bucket: Public-Key Cryptography Resilient to Continual Memory Leakage”. In: *FOCS*. IEEE Computer Society Press, 2010, pp. 501–510. DOI: 10.1109/FOCS.2010.55.
- [BS97] E. Biham and A. Shamir. “Differential Fault Analysis of Secret Key Cryptosystems”. In: *CRYPTO’97*. LNCS. Springer, 1997, pp. 513–525. DOI: 10.1007/BFb0052259.
- [Cas+20] G. Cassiers, B. Grégoire, I. Levi, and F.-X. Standaert. *Hardware Private Circuits: From Trivial Composition to Full Verification*. Cryptology ePrint Archive, Report 2020/185. 2020.
- [CGV14] J. Coron, J. Großschädl, and P. K. Vadnala. “Secure Conversion between Boolean and Arithmetic Masking of Any Order”. In: *CHES 2014*. Springer, 2014.
- [CGZ19] J.-S. Coron, A. Greuet, and R. Zeitoun. *Side-channel Masking with Pseudo-Random Generator*. Cryptology ePrint Archive, Report 2019/1106. 2019.
- [Cha+99] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi. “Towards Sound Approaches to Counteract Power-Analysis Attacks”. In: *CRYPTO*. LNCS. Springer, 1999, pp. 398–412. DOI: 10.1007/3-540-48405-1_26.
- [CK10] J.-S. Coron and I. Kizhvatov. “Analysis and Improvement of the Random Delay Countermeasure of CHES 2009”. In: *CHES*. LNCS. Springer, 2010, pp. 95–109. DOI: 10.1007/978-3-642-15031-9_7.
- [Cnu+15] T. D. Cnudde, B. Bilgin, O. Reparaz, V. Nikov, and S. Nikova. “Higher-Order Threshold Implementation of the AES S-Box”. In: *CARDIS*. 2015, pp. 259–272. DOI: 10.1007/978-3-319-31271-2_16.
- [Cnu+16] T. D. Cnudde, O. Reparaz, B. Bilgin, S. Nikova, V. Nikov, and V. Rijmen. “Masking AES with $d+1$ Shares in Hardware”. In: *CHES*. LNCS. Springer, 2016, pp. 194–212. DOI: 10.1007/978-3-662-53140-2_10.
- [Cnu+17] T. D. Cnudde, B. Bilgin, B. Gierlichs, V. Nikov, S. Nikova, and V. Rijmen. “Does Coupling Affect the Security of Masked Implementations?” In: LNCS. 2017, pp. 1–18. DOI: 10.1007/978-3-319-64647-3_1.
- [Coo+13] J. Cooper, E. D. Mulder, G. Goodwill, J. Jaffe, G. Kenworthy, and P. Rohatgi. *Test Vector Leakage Assessment (TVLA) Methodology in Practice (Extended Abstract)*. ICMC. 2013.

-
- [Cor+12] J. Coron, C. Giraud, E. Prouff, S. Renner, M. Rivain, and P. K. Vadnala. “Conversion of Security Proofs from One Leakage Model to Another: A New Issue”. In: *COSADE*. LNCS. Springer, 2012, pp. 69–81. DOI: 10.1007/978-3-642-29912-4_6.
- [Cor+13] J. Coron, E. Prouff, M. Rivain, and T. Roche. “Higher-Order Side Channel Security and Mask Refreshing”. In: *FSE 2013*. LNCS. Springer, 2013, pp. 410–424. DOI: 10.1007/978-3-662-43933-3_21.
- [Cor+15] J. Coron, J. Großschädl, M. Tibouchi, and P. K. Vadnala. “Conversion from Arithmetic to Boolean Masking with Logarithmic Complexity”. In: *FSE*. Springer, 2015, pp. 130–149. DOI: 10.1007/978-3-662-48116-5_7.
- [Cor17] J. Coron. “High-Order Conversion from Boolean to Arithmetic Masking”. In: *TCHES*. Springer, 2017, pp. 93–114. DOI: 10.1007/978-3-319-66787-4_5.
- [CRR03] S. Chari, J. R. Rao, and P. Rohatgi. “Template Attacks”. In: *CHES*. LNCS. Springer, 2003, pp. 13–28. DOI: 10.1007/3-540-36400-5_3.
- [D’A+17] J.-P. D’Anvers, A. Karmakar, S. S. Roy, and F. Vercauteren. *SABER: Module-LWR based KEM*. Tech. rep. National Institute of Standards and Technology, 2017.
- [D’A+18] J.-P. D’Anvers, A. Karmakar, S. S. Roy, and F. Vercauteren. “Saber: Module-LWR Based Key Exchange, CPA-Secure Encryption and CCA-Secure KEM”. In: *AFRICACRYPT*. LNCS. Springer, 2018, pp. 282–305. DOI: 10.1007/978-3-319-89339-6_16.
- [DBR19] L. De Meyer, B. Bilgin, and O. Reparaz. “Consolidating Security Notions in Hardware Masking”. In: *TCHES 3* (2019), pp. 119–147. DOI: 10.13154/tches.v2019.i3.119-147.
- [DDF14] A. Duc, S. Dziembowski, and S. Faust. “Unifying Leakage Models: From Probing Attacks to Noisy Leakage”. In: *EUROCRYPT*. LNCS. Springer, 2014, pp. 423–440. DOI: 10.1007/978-3-642-55220-5_24.
- [Deb12] B. Debraize. “Efficient and Provably Secure Methods for Switching from Arithmetic to Boolean Masking”. In: *CHES*. Springer, 2012, pp. 107–121. DOI: 10.1007/978-3-642-33027-8_7.
- [Des] *Data Encryption Standard*. National Bureau of Standards, NBS FIPS PUB 46, U.S. Department of Commerce. 1977.
- [DF12] S. Dziembowski and S. Faust. “Leakage-Resilient Circuits without Computational Assumptions”. In: *TCC*. LNCS. Springer, 2012, pp. 230–247. DOI: 10.1007/978-3-642-28914-9_13.

- [DFS15a] A. Duc, S. Faust, and F.-X. Standaert. “Making Masking Security Proofs Concrete - Or How to Evaluate the Security of Any Leaking Device”. In: *EUROCRYPT*. LNCS. Springer, 2015, pp. 401–429. DOI: 10.1007/978-3-662-46800-5_16.
- [DFS15b] S. Dziembowski, S. Faust, and M. Skorski. “Noisy Leakage Revisited”. In: *EUROCRYPT*. LNCS. Springer, 2015, pp. 159–188. DOI: 10.1007/978-3-662-46803-6_6.
- [DH76] W. Diffie and M. E. Hellman. “New Directions in Cryptography”. In: *IEEE Transactions on Information Theory* 6 (1976), pp. 644–654.
- [Din+17] J. Ding, T. Takagi, X. Gao, and Y. Wang. *DingKeyExchange*. Tech. rep. National Institute of Standards and Technology, 2017.
- [DKL09] Y. Dodis, Y. T. Kalai, and S. Lovett. “On cryptography with auxiliary input”. In: *ACM STOC*. ACM Press, 2009, pp. 621–630. DOI: 10.1145/1536414.1536498.
- [DLW06] G. Di Crescenzo, R. J. Lipton, and S. Walfish. “Perfectly Secure Password Protocols in the Bounded Retrieval Model”. In: *TCC*. LNCS. Springer, 2006, pp. 225–244. DOI: 10.1007/11681878_12.
- [DNR19] S. Dhooghe, S. Nikova, and V. Rijmen. “Threshold Implementations in the Robust Probing Model”. In: *ACM Workshop on Theory of Implementation Security*. 2019, pp. 30–37. DOI: 10.1145/3338467.3358949.
- [Dod+10] Y. Dodis, K. Haralambiev, A. López-Alt, and D. Wichs. “Cryptography against Continuous Memory Attacks”. In: *FOCS*. IEEE Computer Society Press, 2010, pp. 511–520. DOI: 10.1109/FOCS.2010.56.
- [DRB18] L. De Meyer, O. Reparaz, and B. Bilgin. “Multiplicative Masking for AES in Hardware”. In: *TCHES* 3 (2018), pp. 431–468. DOI: 10.13154/tches.v2018.i3.431-468.
- [Dwo15] M. J. Dworkin. *SHA-3 standard: Permutation-based hash and extendable-output functions*. Tech. rep. 2015.
- [Dzi06] S. Dziembowski. “Intrusion-Resilience Via the Bounded-Storage Model”. In: *TCC*. LNCS. Springer, 2006, pp. 207–224. DOI: 10.1007/11681878_11.
- [EWS14] H. Eldib, C. Wang, and P. Schaumont. “Formal Verification of Software Countermeasures against Side-Channel Attacks”. In: *ACM* 2 (2014), 11:1–11:24. DOI: 10.1145/2685616.

- [Fau+18] S. Faust, V. Grosso, S. M. D. Pozo, C. Paglialonga, and F.-X. Standaert. “Composable Masking Schemes in the Presence of Physical Defaults & the Robust Probing Model”. In: *TCHES* 3 (2018), pp. 89–120. DOI: 10.13154/tches.v2018.i3.89–120.
- [FG05] W. Fischer and B. M. Gammel. “Masking at Gate Level in the Presence of Glitches”. In: *CHES*. 2005, pp. 187–200. DOI: 10.1007/11545262_14.
- [FJP14] L. D. Feo, D. Jao, and J. Plût. “Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies”. In: *Journal of Mathematical Cryptology* 3 (2014), pp. 209–247. DOI: 10.1515/jmc-2012-0015.
- [FPS17] S. Faust, C. Paglialonga, and T. Schneider. “Amortizing Randomness Complexity in Private Circuits”. In: *ASIACRYPT*. LNCS. Springer, 2017, pp. 781–810. DOI: 10.1007/978-3-319-70694-8_27.
- [Fum+10] G. Fumaroli, A. Martinelli, E. Prouff, and M. Rivain. “Affine Masking against Higher-Order Side Channel Analysis”. In: *SAC*. LNCS. Springer, 2010, pp. 262–280. DOI: 10.1007/978-3-642-19574-7_18.
- [GIB18] H. Gross, R. Iusupov, and R. Bloem. “Generic Low-Latency Masking in Hardware”. In: *TCHES* 2 (2018), pp. 1–21. DOI: 10.13154/tches.v2018.i2.1–21.
- [Gie+08] B. Gierlichs, L. Batina, P. Tuyls, and B. Preneel. “Mutual Information Analysis”. In: *CHES*. LNCS. Springer, 2008, pp. 426–442. DOI: 10.1007/978-3-540-85053-3_27.
- [GM11] L. Goubin and A. Martinelli. “Protecting AES with Shamir’s Secret Sharing Scheme”. In: *CHES*. LNCS. Springer, 2011, pp. 79–94. DOI: 10.1007/978-3-642-23951-9_6.
- [GM17] H. Gross and S. Mangard. “Reconciling d+1 Masking in Hardware and Software”. In: *CHES*. Lecture Notes in Computer Science. Springer, 2017, pp. 115–136. DOI: 10.1007/978-3-319-66787-4_6.
- [GM82] S. Goldwasser and S. Micali. “Probabilistic Encryption and How to Play Mental Poker Keeping Secret All Partial Information”. In: *ACM STOC*. ACM Press, 1982, pp. 365–377. DOI: 10.1145/800070.802212.
- [GM84] S. Goldwasser and S. Micali. “Probabilistic Encryption”. In: *Journal of Computer and System Sciences* 2 (1984), pp. 270–299. DOI: 10.1016/0022-0000(84)90070-9.
- [GMK16] H. Gross, S. Mangard, and T. Korak. *Domain-Oriented Masking: Compact Masked Hardware Implementations with Arbitrary Protection Order*. Cryptology ePrint Archive, Report 2016/486. 2016.

- [GMK17] H. Gross, S. Mangard, and T. Korak. “An Efficient Side-Channel Protected AES Implementation with Arbitrary Protection Order”. In: *CT-RSA*. LNCS. Springer, 2017, pp. 95–112. DOI: 10.1007/978-3-319-52153-4_6.
- [GMR84] S. Goldwasser, S. Micali, and R. L. Rivest. “A “Paradoxical” Solution to the Signature Problem (Abstract) (Impromptu Talk)”. In: *CRYPTO*. LNCS. Springer, 1984, p. 467.
- [GMR88] S. Goldwasser, S. Micali, and R. L. Rivest. “A Digital Signature Scheme Secure Against Adaptive Chosen-message Attacks”. In: *SIAM Journal on Computing* 2 (1988), pp. 281–308. DOI: 10.1137/0217017.
- [Goo+11] G. Goodwill, B. Jun, J. Jaffe, and P. Rohatgi. *A testing methodology for side channel resistance validation*. 2011.
- [Gou01] L. Goubin. “A Sound Method for Switching between Boolean and Arithmetic Masking”. In: *CHES*. Springer, 2001, pp. 3–15.
- [GP99] L. Goubin and J. Patarin. “DES and Differential Power Analysis (The “Duplication” Method)”. In: *CHES*. LNCS. Springer, 1999, pp. 158–172. DOI: 10.1007/3-540-48059-5_15.
- [GPS14] V. Grosso, E. Prouff, and F. Standaert. “Efficient Masked S-Boxes Processing - A Step Forward -”. In: *AFRICACRYPT*. LNCS. Springer, 2014, pp. 251–266. DOI: 10.1007/978-3-319-06734-6_16.
- [GR12] S. Goldwasser and G. N. Rothblum. “How to Compute in the Presence of Leakage”. In: *FOCS*. IEEE Computer Society Press, 2012, pp. 31–40. DOI: 10.1109/FOCS.2012.34.
- [HT19] M. Hutter and M. Tunstall. “Constant-time higher-order Boolean-to-arithmetic masking”. In: *Journal of Cryptographic Engineering* (2019), pp. 173–184.
- [IBM17] IBM Press Release. *IBM Announces Advances to IBM Quantum Systems & Ecosystem*. <https://www-03.ibm.com/press/us/en/pressrelease/53374.wss>. 2017.
- [Ish+06] Y. Ishai, M. Prabhakaran, A. Sahai, and D. Wagner. “Private Circuits II: Keeping Secrets in Tamperable Circuits”. In: *EUROCRYPT*. LNCS. Springer, 2006, pp. 308–327. DOI: 10.1007/11761679_19.
- [Ish+13] Y. Ishai, E. Kushilevitz, X. Li, R. Ostrovsky, M. Prabhakaran, A. Sahai, and D. Zuckerman. “Robust Pseudorandom Generators”. In: *ICALP*. LNCS. Springer, 2013, pp. 576–588. DOI: 10.1007/978-3-642-39206-1_49.
- [ISW03] Y. Ishai, A. Sahai, and D. Wagner. “Private Circuits: Securing Hardware against Probing Attacks”. In: *CRYPTO*. LNCS. Springer, 2003, pp. 463–481. DOI: 10.1007/978-3-540-45146-4_27.

- [Kah96] D. Kahn. *The Codebreakers: The comprehensive history of secret communication from ancient times to the internet*. Simon and Schuster, 1996.
- [Kel18] J. Kelly. *A Preview of Bristlecone, Google’s New Quantum Processor*. <https://ai.googleblog.com/2018/03/a-preview-of-bristlecone-googles-new.html>. 2018.
- [KJJ01] P. C. Kocher, J. M. Jaffe, and B. C. Jun. *Using unpredictable information to minimize leakage from smartcards and other cryptosystems*. US Patent 6,327,661. 2001.
- [KJJ99] P. C. Kocher, J. Jaffe, and B. Jun. “Differential Power Analysis”. In: *CRYPTO*. LNCS. Springer, 1999, pp. 388–397. DOI: 10.1007/3-540-48405-1_25.
- [KL14] J. Katz and Y. Lindell. *Introduction to modern cryptography*. Chapman and Hall/CRC, 2014.
- [Koc96] P. C. Kocher. “Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems”. In: *CRYPTO*. LNCS. Springer, 1996, pp. 104–113. DOI: 10.1007/3-540-68697-5_9.
- [KR19] Y. T. Kalai and L. Reyzin. “A survey of leakage-resilient cryptography”. In: *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*. 2019, pp. 727–794. DOI: 10.1145/3335741.3335768.
- [KRJ14] M. Karroumi, B. Richard, and M. Joye. “Addition with Blinded Operands”. In: *COSADE*. LNCS. 2014, pp. 41–55. DOI: 10.1007/978-3-319-10175-0_4.
- [Med+10] M. Medwed, F.-X. Standaert, J. Großschädl, and F. Regazzoni. “Fresh Re-keying: Security against Side-Channel and Fault Attacks for Low-Cost Devices”. In: *AFRICACRYPT*. LNCS. Springer, 2010, pp. 279–296.
- [Mes00] T. S. Messerges. “Using Second-Order Power Analysis to Attack DPA Resistant Software”. In: *CHES*. LNCS. Springer, 2000, pp. 238–251. DOI: 10.1007/3-540-44499-8_19.
- [Moo+19] T. Moos, A. Moradi, T. Schneider, and F.-X. Standaert. “Glitch-Resistant Masking Revisited”. In: *TCHES* (2019), pp. 256–292.
- [MOP08] S. Mangard, E. Oswald, and T. Popp. *Power analysis attacks: Revealing the secrets of smart cards*. Springer Science & Business Media, 2008.
- [Mos15] M. Mosca. *Cybersecurity in an Era with Quantum Computers: Will We Be Ready?* Cryptology ePrint Archive, Report 2015/1075. 2015.

- [MPG05] S. Mangard, T. Popp, and B. M. Gammel. “Side-Channel Leakage of Masked CMOS Gates”. In: *CT-RSA*. LNCS. Springer, 2005, pp. 351–365. DOI: 10.1007/978-3-540-30574-3_24.
- [MPO05] S. Mangard, N. Pramstaller, and E. Oswald. “Successfully Attacking Masked AES Hardware Implementations”. In: *CHES*. LNCS. Springer, 2005, pp. 157–171. DOI: 10.1007/11545262_12.
- [MR04] S. Micali and L. Reyzin. “Physically Observable Cryptography (Extended Abstract)”. In: *TCC*. LNCS. Springer, 2004, pp. 278–296. DOI: 10.1007/978-3-540-24638-1_16.
- [MW15] A. Moradi and A. Wild. “Assessment of Hiding the Higher-Order Leakages in Hardware - What Are the Achievements Versus Overheads?”. In: *CHES*. Lecture Notes in Computer Science. Springer, 2015, pp. 453–474. DOI: 10.1007/978-3-662-48324-4_23.
- [NIS16] NIST. “Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process”. In: *National Institute of Standards and Technology* (2016).
- [NRS11] S. Nikova, V. Rijmen, and M. Schl  ffer. “Secure Hardware Implementation of Nonlinear Functions in the Presence of Glitches”. In: *Journal of Cryptology* 2 (2011), pp. 292–321. DOI: 10.1007/s00145-010-9085-7.
- [Ode+18] T. Oder, T. Schneider, T. P  ppelmann, and T. G  neysu. “Practical CCA2-Secure and Masked Ring-LWE Implementation”. In: *TCHES* (2018), pp. 142–174. DOI: 10.13154/tches.v2018.i1.142-174.
- [Pei14] C. Peikert. “Lattice Cryptography for the Internet”. In: *PQCRYPTO*. 2014, pp. 197–219. DOI: 10.1007/978-3-319-11659-4_12.
- [Pos+11] A. Poschmann, A. Moradi, K. Khoo, C.-W. Lim, H. Wang, and S. Ling. “Side-Channel Resistant Crypto for Less than 2,300 GE”. In: *Journal of Cryptology* 2 (2011), pp. 322–345. DOI: 10.1007/s00145-010-9086-6.
- [PR11] E. Prouff and T. Roche. “Higher-Order Glitches Free Implementation of the AES Using Secure Multi-party Computation Protocols”. In: *CHES*. LNCS. Springer, 2011, pp. 63–78. DOI: 10.1007/978-3-642-23951-9_5.
- [PR13] E. Prouff and M. Rivain. “Masking against Side-Channel Attacks: A Formal Security Proof”. In: *EUROCRYPT*. LNCS. Springer, 2013, pp. 142–159. DOI: 10.1007/978-3-642-38348-9_9.
- [RD01] V. Rijmen and J. Daemen. “Advanced encryption standard”. In: *Proceedings of Federal Information Processing Standards Publications, National Institute of Standards and Technology* (2001), pp. 19–22.

- [Reg05] O. Regev. “On lattices, learning with errors, random linear codes, and cryptography”. In: *ACM STOC*. ACM Press, 2005, pp. 84–93. DOI: 10.1145/1060590.1060603.
- [Rep15] O. Reparaz. *A note on the security of Higher-Order Threshold Implementations*. Cryptology ePrint Archive, Report 2015/001. 2015.
- [Rep+15] O. Reparaz, B. Bilgin, S. Nikova, B. Gierlichs, and I. Verbauwhede. “Consolidating Masking Schemes”. In: *CRYPTO*. LNCS. Springer, 2015, pp. 764–783. DOI: 10.1007/978-3-662-47989-6_37.
- [Rep16] O. Reparaz. “Detecting Flawed Masking Schemes with Leakage Detection Tests”. In: *FSE*. LNCS. Springer, 2016, pp. 204–222. DOI: 10.1007/978-3-662-52993-5_11.
- [Roy+15] D. B. Roy, S. Bhasin, S. Guilley, J. Danger, and D. Mukhopadhyay. “From theory to practice of private circuit: A cautionary note”. In: *ICCD*. IEEE Computer Society, 2015, pp. 296–303. DOI: 10.1109/ICCD.2015.7357117.
- [RP10] M. Rivain and E. Prouff. “Provably Secure Higher-Order Masking of AES”. In: *CHES*. LNCS. Springer, 2010, pp. 413–427. DOI: 10.1007/978-3-642-15031-9_28.
- [Saa17] M.-J. O. Saarinen. *HILA5*. Tech. rep. National Institute of Standards and Technology, 2017.
- [Sch+19] T. Schneider, C. Paglialonga, T. Oder, and T. Güneysu. “Efficiently Masking Binomial Sampling at Arbitrary Orders for Lattice-Based Crypto”. In: *PKC*. LNCS. Springer, 2019, pp. 534–564. DOI: 10.1007/978-3-030-17259-6_18.
- [Sha00] A. Shamir. “Protecting Smart Cards from Passive Power Analysis with Detached Power Supplies”. In: *CHES*. LNCS. Springer, 2000, pp. 71–77. DOI: 10.1007/3-540-44499-8_5.
- [Sha79] A. Shamir. “How to Share a Secret”. In: *Communications of the Association for Computing Machinery* 11 (1979), pp. 612–613. DOI: 10.1145/359168.359176.
- [Sho97] P. W. Shor. “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer”. In: *SIAM Journal on Computing* (1997), pp. 1484–1509. DOI: 10.1137/S0036144598347011.
- [Sma+17] N. P. Smart, M. R. Albrecht, Y. Lindell, E. Orsini, V. Osheter, K. G. Paterson, and G. Peer. *LIMA-1.1: A PQC Encryption Scheme*. Tech. rep. National Institute of Standards and Technology, 2017.

- [SMG15] T. Schneider, A. Moradi, and T. Güneysu. “Arithmetic Addition over Boolean Masking - Towards First- and Second-Order Resistance in Hardware”. In: *ACNS*. Springer, 2015, pp. 559–578. DOI: 10.1007/978-3-319-28166-7\27.
- [SMY09] F. Standaert, T. Malkin, and M. Yung. “A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks”. In: *EUROCRYPT 2009*. LNCS. Springer, 2009, pp. 443–461. DOI: 10.1007/978-3-642-01001-9_26.
- [ST17] N. I. of Standards and Technology. *Post-Quantum Cryptography - Round 1 Submissions*. <https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions>. 2017.
- [Sta17] F. Standaert. *How (not) to Use Welch’s T-test in Side-Channel Security Evaluations*. Cryptology ePrint Archive, Report 2017/138. 2017.
- [Tou+16] A. de Touzalin, C. Marcus, F. Heijman, I. Cirac, R. Murray, and T. Calarco. *Quantum Manifesto—A New Era of Technology*. <http://quopoe.eu/manifesto>. 2016.
- [Vey+12] N. Veyrat-Charvillon, M. Medwed, S. Kerckhof, and F.-X. Standaert. “Shuffling against Side-Channel Attacks: A Comprehensive Study with Cautionary Note”. In: *ASIACRYPT*. LNCS. Springer, 2012, pp. 740–757. DOI: 10.1007/978-3-642-34961-4_44.
- [von01] M. von Willich. “A Technique with an Information-Theoretic Basis for Protecting Secret Data from Differential Power Attacks”. In: *IMA International Conference on Cryptography and Coding*. LNCS. Springer, 2001, pp. 44–62. DOI: 10.1007/3-540-45325-3_6.
- [Wan+16] W. Wang, F.-X. Standaert, Y. Yu, S. Pu, L. Junrong, Z. Guo, and D. Gu. *Inner Product Masking for Bitslice Ciphers and Security Order Amplification for Linear Leakages*. CARDIS. Springer, 2016. DOI: 10.1007/978-3-319-54669-8\11.
- [WH17] Y. Won and D. Han. “Efficient Conversion Method from Arithmetic to Boolean Masking in Constrained Devices”. In: *COSADE*. Springer, 2017, pp. 120–137. DOI: 10.1007/978-3-319-64647-3_8.
- [WM18] F. Wegener and A. Moradi. “A First-Order SCA Resistant AES Without Fresh Randomness”. In: LNCS. 2018, pp. 245–262. DOI: 10.1007/978-3-319-89641-0_14.